Frankie Robertson

Word Sense Disambiguation for Finnish with an Application to Language Learning

Master's Thesis in Information Technology

March 17, 2020

University of Jyväskylä

Faculty of Information Technology

Author: Frankie Robertson

Contact information: frankie.ray.robertson@student.jyu.fi, frankie@robertson.name

Supervisor: Michael Cochez

Title: Word Sense Disambiguation for Finnish with an Application to Language Learning

Työn nimi: Saneiden alamerkitysten yksiselitteistäminen suomen kielelle ja sen soveltaminen kielen oppimiseen

Project: Master's Thesis

Study line: Mathematical Information Technology

Page count: 191+0

Abstract: The task of automatically determining the correct meaning of a word within some natural language utterance is referred to as Word Sense Disambiguation (WSD). This thesis describes the implementation and evaluation of WSD for the Finnish language, motivated by its novel application to Computer Aided Language Learning (CALL). To serve as training data for Machine Learning (ML) based WSD techniques, a sense-annotated corpus is automatically created based on a collection of bilingual subtitles. Next, several WSD algorithms are adapted to Finnish and evaluated according to their F_1-measure. Then, a Lexical Knowledge Base (LKB) is constructed by clustering and aligning existing resources, and tools to extract and analyse complex lexical units are created. Finally, TheWhatNow?!, a CALL tool which uses WSD on this new lexical resource to offer in context help related to word structure and meaning to language learners is introduced and the design principles guiding its construction and user interface are expounded.

Keywords: Finnish language, Word sense disambiguation, Lexical semantics, Computer aided language learning, Morphology

Suomenkielinen tiivistelmä: Tehtävää sanan oikean merkityksen määritämiseksi automattisesti jossakin luonnollisen kielen ilmaisussa kutsutaan saneiden alamerkitysten yksiselitteistämiseksi. Tämä pro gradu -tutkielma kuvaa saneiden alamerkitysten yksiselitteistämisen toimeenpanoa ja arviointia suomen kielelle, ja sitä motivoi tämän tehtävän uudenlainen soveltaminen tietokoneavusteiseen kielen oppimiseen. Tutkielmassa kaksikieliseen tekstitysaineistoon pohjaava sanojen alamerkitysten mukaan annotoitu korpus on luotu automattisesti palvelemaan opetusaineistona koneoppimiseen pohjautuville saneiden alamerkitysten yksiselitteistämisen tekniikoille. Seuravaksi saneiden alamerkitysten yksiselitteistämisen algoritmeja on muokattu suomen kielelle ja arvioitu niiden F_1-mitan mukaan. Sen jälkeen on rakennettu sekä leksikaalinen tietämyskanta klusteroimalla ja tunnistamalla vastaavuuksia että välineet kompleksisten lekseemien poimimiseen ja analysointiin. Lopuksi on esitelty NiinMikäOli?!, tietokoneavusteinen kielen oppimisen väline, joka käyttää saneiden alamerkitysten yksiselitteistämistä uudella leksikaalisella resurssilla tarjotakseen sanojen rakenteeseen ja merkitykseen liittyvää kontekstisidonaista apua kielenoppijoille. Lisäksi on selitetty NiinMikäOli?!:n rakentamista ja käyttöliittymää ohjaavat suunnittelun periaatteet.

Avainsanat: Suomen kieli, Saneiden alamerkitysten yksiselitteistäminen, Sanasemantiikka, Tietokoneavusteinen kielen oppiminen, Morfologia

List of Figures

List of Tables

Contents

Glossary

allomorph Any of two or more actual representations of a morpheme. 140, 145

clamping An operation which ensures a value falls within a specified range by moving it to the nearest value within the available range:
$\text{clamp}(x, range_{min}, range_{max}) = \text{min}(\text{max}(x, range_{min}), range_{max})$. 122, 123

clique A complete subgraph. 80, 113

complete graph A graph where for all $v_1, v_2 \in V$, $(v_1, v_2) \in E$. 1, 112

confusion class A set of possible values which a prediction procedure may choose from for a given input. 38–40, 68, 69

corpus A large, structured set of texts. i, iii, iv, vi, vii, 10, 15, 16, 23, 24, 27, 30, 32, 33, 35, 36, 38–41, 43–45, 49, 50, 54, 57, 58, 61, 66–71, 73, 75, 76, 81, 83, 85, 86, 91–93, 96, 98, 99, 132, 155, 157, 159

directed graph A graph where for $e \in E$, $v_1, v_2 \in V$, the edges have the form $e = (v_1, v_2)$. 19

enclitic A word-like unit which barely any stress in speech, and is attached on to the previous word in text. 18, 108, 137

fixed point Given a domain D and a function $f\colon D \to D$, a fixed point is a value $p \in D$ for which $f(p) = p$. A value $x \in D$ is said to reach its fixed point when we obtain a fixed point $x' \in D$ by repeated application of f: $x' = f(\ldots f(x)\ldots)$. 47

gloss A translation, explanation, interpretation, or paraphrase of a piece of text. Usually a gloss has the same POS and semantics as the unit which is glosses. 7, 27–29, 33, 81–83, 123, 124, 126

graph A tuple (V, E), where V is a set of vertices or nodes and E is a set of edges connecting them. iii, 1, 2, 17, 20, 27, 29, 39, 80, 81, 94, 112, 120, 121, 126, 141, 142

L1 In Second Language Acquisition (SLA): A subject's first or native language. 36

L2 In SLA: A subject's second or target language. 34, 36

lexicon The vocabulary of a person or language; The abstract platonic ideal of the set of those units (e.g., words, lemmas, or morphemes) which hold meaning in a language. 19, 40, 45, 134

overfitting When a ML procedure learns too much about specific features of its training data that result from the sampling process, and not enough features about the whole distribution underlying the training data. 75, 95

Pareto front Given dimensions $x_1, x_2, ..x_n$ each equipped with an order where $x \geq y$ can be read as x is preferable or equal to y, the Pareto front is that set of points for which for each member x there exists no solution x' in which for every dimension is preferable, $x'_i \geq x_i$. 61, 62, 65, 128

particle A word which has a grammatical function, but no definition in and of itself. 18, 53, 108, 137

partition Given a set S, its partition is a set P of nonempty subsets of S, such that each member of S is a member of exactly one member of P. vii, 2, 19, 112, 115

pipeline A series of data processing stages operating upon data streams, where the output of each feeds into the input of the next. The processes may be executed in parallel. iii, 44, 46, 50, 51, 54–56, 76, 102, 110, 153

scraping A procedure which extracts structured data from unstructured or semi-structured data. vii, 100, 103, 106, 136, 151, 156

smoothing In the context of estimating probabilities based on count data: any procedure which aims to adjust probabilities to account for unseen data. 50, 82, 108, 132

subgraph Given a graph (V_1, E_1), another graph (V_2, E_2) is a subgraph when $V_2 \subseteq V_1$ and $E_2 \subseteq E_1$. 1, 29, 30

substantive In descriptions of Finnish grammar: any word to which a case ending can be attached, including all nouns and adjectives. 108

undirected graph A graph where for $e \in E$, $v_1, v_2 \in V$, the edges have the form $e = v_1, v_2$. 80

wildcard Some part of a matching specification which is able to match anything in the domain of matchable objects freely e.g., * is commonly used as a wildcard character. 3, 109–111, 139, 147

Acronyms

AI Artificial Intelligence 6, 7, 19

AWE Average/Aggregate of Word Embeddings iv, vii, 29, 31, 71, 83–86, 89–91, 95

BLEU Bi-Lingual Evaluation Understudy 13, 14

BOW Bag Of Words 28, 31

BPE Byte Pair Encoding 85, 98, 134

CALL Computer Aided Language Learning i, iii, vi, 6–8, 17, 33, 35

EMD Earth Mover's Distance 124, 125

ETL Extract Transform Load iv, 150, 153, 154

FiPB Finnish PropBank v, 22, 113, 115, 116

FiWN FinnWordNet iii, v, 21, 47–50, 53, 56–58, 60, 67, 68, 71, 72, 99, 109, 111, 113, 115–117, 122, 153, 156–158

FN False Negative 78, 80, 103, 106, 119, 120, 122, 126

FP False Positive 46, 78, 80, 103, 106, 119, 120, 126

FST Finite-State Transducer 110

HTML HyperText Markup Language 100

JSON JavaScript Object Notation iii, 102–104, 148, 151

LKB Lexical Knowledge Base i, vii, 6, 16, 21, 22, 29, 30, 48, 56, 67, 68, 80, 81, 99, 132

LSTM Long Short-Term Memory 25, 26, 84

ML Machine Learning i, 2, 15, 26, 80

MT Machine Translation 9–11, 13, 14

MWE Multi-Word Expression iii, vii, viii, 46–48, 74, 99, 103, 108, 109, 130, 132, 134, 146, 147, 151, 153, 154, 156–158

NLP Natural Language Processing vi, 9–11, 20, 67, 80, 155

NMT Neural Machine Translation 14

NN Nearest Neighbour iv, vii, 83–85, 89–91, 95

PBMT Phrase Based (statistical) Machine Translation 9, 10, 14, 25

POS Part Of Speech iii, 1, 9, 18, 32, 34, 36, 47, 52, 53, 58, 68, 69, 71–73, 76, 83, 100, 134, 138–140

PWN Princeton WordNet iii, 21, 22, 49, 68, 71, 72, 99, 139, 153, 157

RBMT Rule Based Machine Translation 14

RNN Recursive Neural Network 25

SLA Second Language Acquisition vi, 1, 6, 17, 33, 34

SVM Support Vector Machine 31, 83

TN True Negative 106, 119, 120

TP True Positive 46, 62, 78, 103, 106, 119, 120

WSD Word Sense Disambiguation i, iii, iv, vi, 6–11, 16, 17, 19, 26–33, 37–39, 48, 49, 54, 57, 66–68, 71, 76, 79–81, 83, 86, 91, 93, 96, 98, 99, 151, 155, 156, 159, 160

XML eXtensible Markup Language 54, 57, 58, 100

1 Introduction

What makes learning a second language hard? How can a computer help? These are central questions in the fields of Second Language Acquisition (SLA) and Computer Aided Language Learning (CALL) respectively. A language acquisition expert could as well ask "what makes learning a second language easy?" After all, 60% of the world speak more than one language, making multilingualism the norm (Richard 1999). How do we reconcile this with the fact that so many language learners, particularly adults, report significant struggles learning a new language and that high attrition rates are reported, both for self-study (Nielson 2011; Mcquillan 1997) and in university level courses (Dupuy and Krashen 1998).

Here, the perspective is taken that a major barrier to language learning is that when language is learnt in an artificial environment, the brain is less likely to retain the information since it is less likely to seem truly relevant or important (Pearce 2016; Schmidt 1990). To move away from the problems of language learning in a synthetic environment, it should be learnt continually, and language help should be given within the contexts where the language learner needs to use the second language.

In this thesis, I describe the motivation, implementation and evaluation of the *TheWhatNow?!* system. TheWhatNow?! is a tool intended to assist language learners with text comprehension. The primary purpose of the tool is to allow learners to look up words while they are reading Finnish language web pages. For a given part of a given text, TheWhatNow?! ranks various information which may be useful to language learners, so as to place the most useful information in the highest ranking position. In the current version of TheWhatNow?! the information is limited to word definitions and word structure.

Ranking word definitions according to how useful they are to a learner is treated as equivalent to ranking them according to how well they define a word usage within a context. This task is referred to as Word Sense Disambiguation (WSD), formally the task of assigning words a sense from a sense inventory, i.e., an electronic dictionary or Lexical Knowledge Base (LKB) (Jurafsky and Martin 2019a; Navigli 2009). Like many natural language understanding tasks, WSD has been referred to as AI-complete (Mallery 1988, p. 57), that is to say, it is considered as hard as the central problems in Artificial Intelligence (AI), such as passing the Turing test (Turing 1950).

Thus, the purpose of this thesis is to implement and evaluate WSD for the Finnish language, motivated by its novel application to CALL. The connection between WSD and CALL is shown in Figure 1. In order to achieve this objective, I approach the following research questions:

RQ1. How can WSD techniques be evaluated? How can they be compared in terms of their approach? What are the main themes in previous WSD research?

RQ2. What steps are required to implement and evaluate the chosen WSD techniques?

RQ3. Comparatively, how well do the chosen WSD techniques work for the Finnish language?

RQ4. What are the issues with presenting Finnish morphology and word definitions to Finnish learners? How do current approaches fall short when used for this purpose? How can an approach towards addressing these shortcomings be implemented?

1.1 Word Sense Disambiguation

To make the task of WSD concrete, consider an illustrative example: we may have a dictionary containing the following word definitions, also referred to as glosses, for the word *bank*:

Bank.1 An institution where one can place and borrow money and take care of financial affairs.

Bank.2 (hydrology) An edge of river, lake, or other watercourse.[1]

Additionally, we may have two sentence fragments: "...he created [a sculpture of a pair of shoes] on the east <u>bank</u> of the Danube River...", and, "[a]n account held in a foreign offshore <u>bank</u>..."[2].

1. Word senses were obtained from Wiktionary.
2. Reproduced from the Wikipedia articles *Shoes on the Danube Bank* (retrieved from `https://en.wikipedia.org/w/index.php?title=Shoes_on_the_Danube_Bank&oldid=932093964`) and *Offshore Bank* (retrieved from `https://en.wikipedia.org/w/index.php?title=Offshore_bank&oldid=928783206`) respectively.

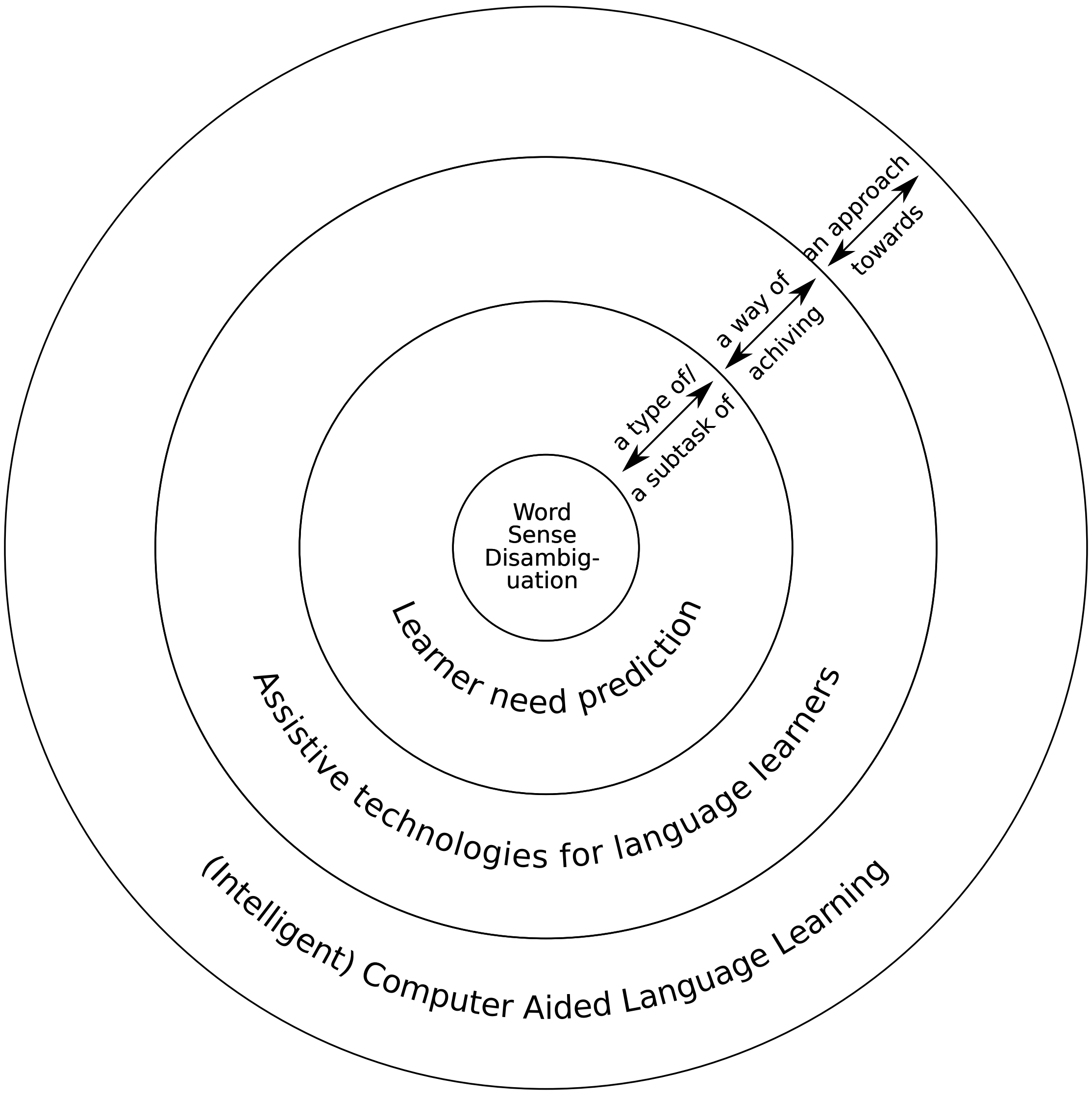

Word Sense Disambig- uation
Learner need prediction
Assistive technologies for language learners
(Intelligent) Computer Aided Language Learning
a type of/ a subtask of
a way of achiving
an approach towards

Figure 1: A conceptual map showing the relationship between WSD and CALL.

As humans, we can assign the correct dictionary sense to these occurrences using various contextual cues. For example, with the first sentence fragment, we can see that the concept referred to by <u>bank</u> is **bank.2** because there are shoes positioned on top of it, which is more likely because **bank.2** refers to a physical thing whereas **bank.1** refers to an abstract thing. Another approach would be to see that it is nearby the word "river", which also appears in the definition of **bank.2**. A third way would be to look only at the two words "east <u>bank</u>" and know from statistical evidence that this is more likely to refer to **bank.2**. In fact, each of these approaches forms the basis for a different family of WSD algorithms.

Beyond any intrinsic novelty or intellectual challenge, why should we care about the task of WSD? Resnik (2006) argues that there have been three types of arguments made for the importance of research into WSD, summarised here:

- *Argument from faith/tradition*: In this argument, the importance of WSD is "passed from teacher to student and easily accepted on intuitive grounds" as "'obviously essential for language understanding applications"'.
- *Argument by analogy*: In this argument, an analogy is made to other tasks and linguistic representations such a Part Of Speech (POS) tagging, the task that involves determining, for example, whether a particular word is a verb or a noun. Since these tasks are used as part of larger tasks and applications, they are referred to as intermediate tasks. It is then assumed that treating WSD as an intermediate task will be beneficial by analogy.
- *Argument from specific applications*: The previous two arguments fail to convince us that performing WSD will ultimately have any practical benefit. This last argument states that WSD is useful if we can find a specific application where a system can be improved by performing WSD as an intermediate task.

The first two arguments are usually made because of difficulty successfully constructing the third. That is to say, there is a disconnect between the traditional intuition that WSD is required for various Natural Language Processing (NLP) tasks and the current reality that it is rarely used explicitly as an intermediate task in practice. Here, I briefly explore this disconnect by examining the application area of Machine Translation (MT) and the, until recently, state of the art approach of Phrase Based (statistical) Machine Translation (PBMT). MT was one of the first NLP applications to be explored and as early as 1947, word sense ambiguity was noted as an obstacle to MT — surmountable or not. Warren Weaver wrote in a letter to Norbert Wiener dated 4th March 1947: "Recognizing fully, even though necessarily vaguely, the semantic difficulties because of multiple meanings, etc., I have wondered if it were unthinkable to design a computer which would translate.", to which Wiener replied, "as to the problem of mechanical translation, I frankly am afraid the boundaries of words in different languages are too vague and the emotional and international connotations are too extensive to make any quasi mechanical translation scheme very hopeful.". Furthermore, the oft-told apocryphal story of "The spirit is willing but the flesh is weak" being translated into Russian as "The vodka is good but the meat is rotten" (Hutchins 1995) is arguably a problem of incorrectly assigning the word senses "spirit (alcoholic beverage)" and "flesh (animal)" rather than "spirit (mysticism)" and "flesh (human)" to the original text.

A PBMT system, such as Moses (Koehn et al. 2007), or the version of Google Translate rolled out in April 2006 (Och 2006), finds statistical correspondences between phrases in a bilingual corpus. First, sentences within a bilingual corpus are aligned, e.g., using the Gale-Church alignment algorithm (Gale and Church 1993). Next, words within each pair of sentences are aligned, e.g using an IBM word alignment model (Brown et al. 1993), as implemented within Giza++ (Och and Ney 2003). Finally, chunks for which word alignment supports a phrase-to-phrase correspondence within the aligned sentences are counted to form a phrase table with probabilities. If there is sufficient evidence within an English-Finnish bilingual corpus, we would find that the phrase "river bank" corresponds to "joen penkka", while "bank account" corresponds to "pankkitili". Thus, some degree of WSD is performed, but as a side effect of performing PBMT, rather than as a prerequisite for performing it. Alternatively stated, WSD is needed for the task, but the intermediate representation of disambiguated text is not.

This trend away from human designed intermediate representations, corresponding to the outputs of intermediate tasks, has only accelerated following the increased dominance of end to end neural network based approaches for most NLP tasks, further eroding the argument of analogy with other intermediate tasks. In this thesis, WSD is developed to help with the specific application of TheWhatNow?!. Thus it is the third argument, that a specific application benefits from WSD, that is used to support its development here.

1.2 Evaluation in NLP

This section explores the topic of evaluation in NLP in a wider scope, beyond just the task and frame of WSD, in order to expound the basic theoretical and methodological foundation of this thesis. Here we take MT as an example to demonstrate some of the issues which NLP evaluations must take into account. MT is used as an example since it is a very familiar NLP application which presents a variety of challenges in terms of evaluation. The specifics of evaluation in the context of WSD are discussed in Section 2.3.5.

1.2.1 Framing

Before we can even begin NLP system evaluation, we must address the framing of the task. The frame tells us how the task fits in with the rest of the world. For example, whether it is used as part of a larger application and where the interface with end users ultimately lies. This gives us an idea both of what the evaluation should emphasise and how we can interpret the results.

For example, we might have one MT frame where we are attempting to translate controlled language for the exchange of patient records between countries. In this case, it is very undesirable that any mistakes are made in case they lead to misunderstandings that could ultimately cause incorrect medical decisions to be made. However, we would be helped through the use of controlled language, which may allow us to guarantee that translations are correct, leaving only the potential drawbacks of doctors not being able to express their thoughts fully.

This is very different from the much wider frame of everyday personal usage of MT, such as translating a portion of a website using a service such as Google Translate. In this case, the text may be from any domain and users may come with varying levels of expectations. At one extreme a user may expect an entirely correct and fluent translation. An intermediate expectation could be that of obtaining a non-fluent, mostly comprehensible translation. The minimal expectation would generally be to obtain a translation that is just comprehensible enough to be able to discern the topic or the gist of what is being said.

In this latter type of frame, we usually abandon any attempt to make particular guarantees about our system's output, since we have similarly jettisoned any demand for constraints on our input. We now may encounter any type of text and one translation may be better or worse (or hard-to-say-either-way) than the other, according to one or more criteria. In order to make a quantitative judgement about said criteria, we can take a comparative approach, comparing the output of multiple systems.

1.2.2 Comparative evaluation

In a comparative evaluation, the outputs of multiple systems are compared, often against gold standard inputs and outputs. The gold standard data should act as a representative sample against which one or more hopefully representative metrics are measured. The degree of representativeness is defined as how well it represents the task frame.

When a comparable system is not available, an additional — usually very simple system — may be constructed as a baseline. In some cases, a baseline system does not even attempt to achieve the task dictated by the frame but rather acts mainly as a floor to calibrate the chosen metric(s) to the given data set. Another extreme possibility is to have a human attempt the task and then measure the agreement to the gold standard. This would then give a ceiling on a system's performance on a given metric.

Due to the importance of comparative evaluation, the academic computational linguistics community organises shared evaluations, in which a common frame and evaluation method, including a choice of metrics and data set, is fixed ahead of time. These events have the immediate benefit that we can directly compare several systems, which are documented in system papers presented at the conference of the shared evaluation. They have the additional benefit that they establish a standardised benchmark, which is an attractive target for developers of future systems to include in their publications since it allows them to compare against at least all the systems which were entered into the shared evaluation.

1.2.3 Metrics

An evaluation metric is intended to reflect how well a system performs within some frame. Often, what dictates exactly how well a system performs in terms of the frame may be quite far removed from the system itself, and measuring it may be expensive or otherwise undesirable.

For example, within the frame of the patient record MT system, we might actually want to measure how many negative patient outcomes result from the use of the system. However, this would be expensive and would not be ethical. Usually, we must instead accept some surrogate designed to correlate with what we are truly interested in, and along with it some imperfections.

One metric commonly used in MT, used here as an example of an imperfect metric, is Bi-Lingual Evaluation Understudy (BLEU) (Papineni et al. 2002). We are given a sentence to be scored, which has been machine translated from some source sentence as well as one or more human produced reference translations of the same source sentence. BLEU is then defined as the geometric mean of n-gram precisions[3], with respect to the reference translations, calculated with different sizes of n-grams from 1-grams to 4-grams, with a brevity penalty for short translations:

$$\text{BLEU} = \min(1, \frac{\text{output-length}}{\text{reference-length}})(\prod_{i=1}^{4} \text{precision}^{i})^{\frac{1}{4}}$$

where precision^{i} is the i-gram precision, that is the fraction of true positive i-grams over the fraction of all guessed i-grams. An i-gram is a true positive when it matches any of the reference translations. The brevity penalty is defined in terms of reference-length, which is the length of the shortest reference translation.

An alternative would be to use human judgement of a machine translation. In this case, humans may either rate translations with respect to the sentence in the source language or a reference translation. Translations can be rated, for example on a five point scale, according to *adequacy*, that is how understandable it is, and *fluency*, that is how good the use of language is in terms of grammar and idiomatic word choice.

Why was BLEU created when we can use human judges? A readily apparent reason is that blind human evaluation was too expensive. BLEU is used as a surrogate on the basis that it was shown to correlate with human judgement. Not only is BLEU cheaper, but reference translations can be reused by future systems to benchmark against. To do so with human judges would require either re-rating the output of old systems or suffering the confounding

3. Precision is defined in Section 2.3.5.

effect of comparing scores across a different set of human judges. Following its introduction, BLEU became the standard measure in MT, overseeing the transition from Rule Based Machine Translation (RBMT) to PBMT, many improvements in PBMT, and the subsequent transition to Neural Machine Translation (NMT). However, the nature of BLEU gives many pause. How can you score a translation by just comparing it to one or a few reference translations? The set of good translations of a sentence may be very large, and may include very good translations dissimilar to the references. More practically, as Callison-Burch, Osborne, and Koehn (2006) note, it does not always correlate with human judgement. When compared with human judgements, BLEU systematically penalised RBMT versus PBMT.

While the advantages of BLEU come with a fair share of drawbacks, it nevertheless has allowed people to conduct evaluations in small teams with limited resources. However, BLEU may have come to the end of its usefulness. For example Wu et al. (2016) experiment with a variant of their NMT system which is trained using an optimisation process specifically tuned to the definition of BLEU and discover that while it increases BLEU, it decreases their scores according to human annotators in comparison to their unmodified system. Increasing their BLEU score made their system worse. That said, BLEU scores on standard benchmark datasets are still being given[4] in addition to human evaluators, as an extra evaluation which can be conducted cheaply.

We might consider that the design of almost any evaluation will have to contend with some set of trade offs. Järvinen (2012) formalises a very generalised trade off as the iso-epistemic curve, noting that, particularly when attempting to answer research questions involving humans, we must trade off between rigour versus relevance. In this case, it may be easier to get more rigorous results using an automatic metric, notably we have removed confounding variables by using an evaluation technique which is completely repeatable. However, the relevance of the results obtained this way can certainly be bought into question.

4. See for example Hassan et al. (2018).

1.2.4 Representative sampling

When we perform an evaluation metric based on some gold standard data, the choice of gold standard becomes part of the choice of metric, and the same considerations about representativeness with regards to the frame need to be taken into account. We now introduce the related notions of probability distribution and domain. Statistical learning theory treats all data as being generated by a random process following some probability distribution. The field of linguistics has long characterised different distributions of natural language text in terms of domain. One domain of discourse could be for example talking about a patient's medical history. A coarser grained domain may be the genre of the text, for example, news wire. At the coarsest level of domain, we have dialects and languages themselves.

When we train an Machine Learning (ML) based system on some data set, usually the best case is that it will learn something about the distribution that data set comes from. If the training data is from one or more domains, the resulting system will usually be to some extent adapted to those domains. More broadly we may also have non-ML based systems which are somehow adapted to some particular domain. It is quite common with ML evaluations, that the gold standard will be from the same domain as the training data, since they are often different sections of the same corpus, which may just be a single genre. Due to the domain adaption property mentioned, this will usually show relatively good performance. However, the gold standard data should be chosen to be a representation of the chosen frame rather than out of convenience. If the gold standard's distribution does not represent that of the frame but does match the training data, the evaluation will dramatically overestimate the performance, and may rank systems that show stronger domain adaptation properties above those that generalise better and are thus more suitable for the frame.

Indeed, as mentioned, in many frames it is normal for a user to expect some kind of sensible output for all sorts of inputs. Therefore when a system has been trained on data representing a carefully chosen range of domains, it may still eventually encounter some out of domain text. Thus for these frames, it may be important that some sort of out of domain performance is given, perhaps in addition to in domain performance, since it can help to give a more realistic expectation of a system's performance in both favourable and unfavourable situations.

1.3 Structure

The structure of this thesis is as follows. Chapter 2 provides background and reviews related work, as well as addressing RQ1: "How can WSD techniques be evaluated? How can they be compared in terms of their approach? What are the main themes in previous WSD research?"

Then Chapters 3, 4, 5 and 6 proceed in a bottom up fashion. In each case, the foundations built in the former chapters were required as infrastructure to implement the latter, summarised now in reverse. Chapter 6 presents a reading assistant[5] for learners of Finnish. To increase the amount of material available to the reading assistant, and to reduce redundancy in said material, Chapter 5 describes methods for aligning and clustering word senses from multiple LKBs into a new, larger LKB. These two chapters address RQ4: "What are the issues with presenting Finnish morphology and word definitions to Finnish learners? How do current approaches fall short when used for this purpose? How can an approach towards addressing these shortcomings be implemented?"

To improve the reading assistant so that it can offer the most relevant information, the assistant should perform WSD. Chapter 4 describes the adaption, implementation and evaluation of WSD algorithms for Finnish, addressing RQ2: "What steps are required to implement and evaluate the chosen WSD techniques?" and RQ3: "Comparatively, how well do the chosen WSD techniques work for the Finnish language?" To evaluate the WSD algorithms and to implement supervised forms of WSD, sense tagged corpora were needed, and the process by which these were obtained is given in Chapter 3, addressing RQ2. Finally, Chapter 7 finishes with some concluding remarks and possibilities for future work.

5. See Section 2.4.3.

2 Background

This chapter lays some of the foundations needed for the rest of the thesis. Section 2.1 defines some basic linguistic terms and their relationship to each other. Section 2.2 introduces broad categories of lexical resources, those based on graphs and those based on vector spaces. Section 2.3 builds on this to review Word Sense Disambiguation (WSD) techniques, some of which are based on the resources introduced in Section 2.2. Section 2.2 provides foundational background for Chapters 4 & 5, while Section 2.3 gives specific background for Chapter 4. Finally Section 2.4 reviews some major ideas in Second Language Acquisition (SLA) theory before moving onto cover the major approaches to Computer Aided Language Learning (CALL) which are similar to or related to TheWhatNow?!. It serves as background for Chapter 6.

2.1 Linguistic background & Finnish morphology

This section is based on Robertson (2016). In linguistics, morphology is the study of word forms and word formation (Matthews 1991, p. 3). Within the field of morphology, there are a number of different models corresponding to a variety of different types of analysis which are performed for different purposes or reflect different views of language. There are continuing fundamental debates in linguistics about how best to break down, abstract and analyse language; whether there exist some natural, universal set of definitions and what they might be (Aitchison, 2012, pp. 33-34; Lyons, 1968, pp. 196-197). However, for our purposes the definitions of Matthews (2007), form a reasonable starting point:

- A *morpheme* is a minimal unit of grammar into which a sentence or a word within a sentence can be divided. For example, in Finnish we could analyse “puhun” as made up of the root morpheme “puhu” and the first person singular suffix morpheme “–n”.
- A *word* is the smallest of the units that make up a sentence and is marked as such in writing. For our purposes, we take the definition of a word as that which is delimited by whitespace or punctuation.

- A *lexeme* is a word considered as a lexical unit, in abstraction from the specific forms it takes in specific constructions. One way of thinking of a lexeme is as being a set of all inflections of a dictionary form of a word. The lexeme is identified by its dictionary entry which consists of a *lemma*, also known as a headword, and a homonym ordinal.

Morphological parsing, morphological segmentation or morphological analysis is the task of going from a word to a more structurally rich representation within the above framework. In particular, we usually want to extract the lemma and a series of *tags*, describing the Part Of Speech (POS) (e.g., verb) and a grammatical analysis of how the lexeme has been inflected. In an agglutinative language like Finnish, in contrast to a fusional language like Latin, there is a close relationship between the list of morphemes and the set of tags (Matthews 1991, pp. 107-114). The list of all tags available for inclusion in an analysis and the rules for their formatting form a data format called a tag set. In Finnish, for example, a tag could represent the case (e.g., inessive "-ssa") or the presence of an enclitic particle (e.g., interrogative "-ko") if it is present. Morphological generation is the opposite process: that of going from a lexeme and some set of tags to a word form.

As well as inflection, morphological analysis might need to also consider processes of lexical derivation[1]. This is when new lexemes, with new distinct meanings, are formed from old ones. Here we consider two forms: morphological derivation and word compounding. Morphological derivation is the process of a new lexeme being formed by affixing a morpheme to an existing lexeme, possibly changing the POS in the process. A Finnish example is the verb "ajaa" (en: to drive), and the agent morpheme "-ja" (Karlsson 2015, p. 275) forming the agent noun ''ajaja" (en: driver). Derivational morphemes are described as productive when they can be attached to more words of a particular POS, and result in a word which is understandable and considered valid by a native speaker (Matthews 1991, p. 69). Word compounding occurs when multiple words are attached to form new lexemes. In Finnish, both types of lexical derivation happen in a mutually recursive manner and can mix with inflection. For example, given a compound word, each internal word can be inflected, and can

1. Also referred to as word formation.

have already undergone lexical derivation itself. Consider for example: praha:ssa=käy-mättöm-yys=kompleksi "complex about never having been to Prague"[2].

Derivation is traditionally considered distinct from inflection: inflection is said to be a grammatical phenomenon, whereas derivation is said to occur within the lexicon. It should be mentioned that the distinction is based on multiple criteria and may not always be entirely clear cut. Matthews (1991, p. 43) gives a number of edge cases and arguments against the distinction, gradually bringing in more criteria to deal with the various edge cases. Given the distinction is made on not entirely solid ground, alternative models have been formulated, such as Booij (1996), who presents a tripartite model. An example of an edge case is the English word "sands", which has the sense *desert*, and thus here "-s" behaves more like a derivational morpheme than an inflectional morpheme. In applied linguistics, the distinction can be made on a pragmatic basis: we shouldn't expect to find inflected forms in a normal dictionary[3], but we would expect to find at least the most common derived forms.

2.2 Lexical resources: Lexical Knowledge Bases and embeddings

To successfully perform WSD, we must have some kind of machine readable representation of our word senses. We may also choose to make representations of surface forms or lemmas.

In general, machine knowledge representations can be roughly divided into symbolic "Good old fashioned Artificial Intelligence (AI)" representations, which tend to be (mostly) manually created on the one hand, and on the other hand, continuous or fuzzy representations, which tend to be (mostly) derived from unstructured data which was originally created for a purpose other than creating a lexical resource. This rough division is not an exact partition of the possibilities, with hybridisation of the approaches being possible.

Within the former category, a common representation is that of an edge labelled directed graph, referred to as a *knowledge base*. This representation is also sometimes referred to as an ontology, especially when it is coupled with inference rules (in the style of a deductive or active database) and consistency checking. In this representation, there is some set of rela-

2. ":" indicates inflection, "-" indicates morphological derivation, and "=" indicates compounding. Example reproduced from Karlsson (2015).

3. For Finnish, however, we might expect to find their inflection class.

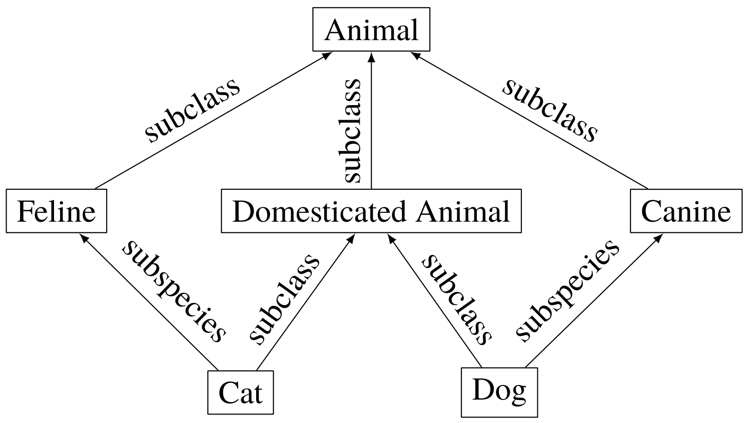


Figure 2: Edge labelled graph representation of the difference and similarities between the concepts of “dog” and “cat”.

Table 1: Table showing a labelled data matrix D representing the differences and similarities between the concepts of “dog” and “cat” in terms of cooccurring attributes.

Entity	Barks	Meows	Can stroke	Likes walks	Likes string	Lives in wild
Cat	0	1	1	0	1	0
Dog	1	0	1	1	0	0

tions that label the directed edges which can connect nodes, typically considered as entities. Usually, entities may also have data attached as named properties, such as a string or number.

Within the latter category, a common representation is an *embedding* of concepts as vectors in a high dimensional vector space. The dimensions may be unlabelled and could represent latent or hidden features, or else the embedding has other desirable features, such as placing similar concepts close to one another.

Figure 2 and Table 1 give an example of how the two representations might capture the similarities and differences between the concepts “dog” and “cat”. In the general case, the cooccurring attributes introduced in Table 1 could be any type of thing which cooccurs with the entity. For example, it could be that we make observations of different cats and dogs and count how many times the different attributes apply. In the context of Natural Language Processing (NLP), the observations can be, for example, counts of other surface forms or lemmas within a fixed-size context window of a token referring to the entity. Notably, these types of cooccurences can be collected automatically without human intervention.

2.2.1 Lexical Knowledge Bases

Princeton WordNet (PWN) (Fellbaum 1998) is a Lexical Knowledge Base (LKB) structured around sets of synonyms, referred to as synsets. Each synset contains many lemmas. The primary relationships are between synsets. Synsets can be related through links such as hypernymy, an is-a type relationship and metonymy, a part-whole relationship. There are also links between lemmas along morpho-semantic lines. So the sense of the verb "drive" which means "to drive a car" is linked to "driver" as in "the driver of the car", but the sense of "drive" as-in "to drive one's self mad" is not.

There have been WordNets created for languages other than English. A common approach is to simply add extra lemmas in the new language to the existing synsets in PWN. FinnWordNet (FiWN) (Lindén and Carlson 2010), the Finnish WordNet, takes a slightly different approach and aligns at the lemma level. SALDO (Borin, Forsberg, and Lönngren 2013), a Swedish LKB, typifies an even more divergent approach, where first an LKB is created without regard to PWN's structure and organising principles, and then as an additional step, an alignment between its word senses and PWN synsets is created. Bond and Paik (2012) created Open-MultiWordNet by finding WordNets under permissive licenses, including FiWN & SALDO, and converting them to a common format.

Apart from WordNet style approaches, another common approach is to create an LKB focussed around predicate-argument structures. A typical example would be a subject-verb-object sentence where the verb specifies and event and the subject and object specify participants "Freddy hit me", which would have the logical predicate-argument structure "hit$_1$(freddy, me)" with hit$_1$ being defined as taking two arguments: the first of type agent: an active participant, and the second of type patient: a passive participant. This family of approaches include FrameNet, (Baker, Fillmore, and Lowe 1998) VerbNet (Schuler 2006) and PropBank, (Palmer, Gildea, and Kingsbury 2005) which is explored further here. The verb senses in PropBank are distinguished as frames or templates of a predicate–argument structure. The underlying assumption is that the basic meaning of the verb, or predicate is determined chiefly by the role its arguments have in relation to it. Here, assuming a predicate refers to an event, its arguments tend to roughly be those other parts of the sentence that specify the "whats" and "hows" in relation to it. The resulting verb senses are more coarse grained than WordNet.

Finnish PropBank (FiPB) (Haverinen et al. 2015) is an LKB based on the original English PropBank (Palmer, Gildea, and Kingsbury 2005). Predicate frames are necessarily quite particular to a language, and thus creating a PropBank for a new language is usually not simply or only a matter of adding new labels in the new language to existing frames. That said, in practice, many frames in FiPB are modelled after those in English PropBank.

Predicate Matrix version 1.3 (Lacalle et al. 2016) builds on top of other resources to create an extensive mapping between, among other LKBs, English PropBank and PWN. No evaluation was performed for Predicate Matrix 1.3, but Lacalle et al. (2016) estimated the precision of the WordNet-PropBank mapping in Predicate Matrix 1.2 as 71.3%.

2.2.2 Embeddings

Embeddings, or vector space models, represent concepts in a high dimensional space. Early work in this area started in the late 50s and early 60s in the application area of information retrieval and many of the ideas from this time were implemented in the SMART document retrieval system project, described in Salton (1971). These early term-document vector space models were sparse, with each element in a document's vector representing either a single term or stem, or to combat data sparsity a manually created grouping of words representing a synonym set or topic. Two techniques of note in the SMART system that are still in use is the use of cosine similarity between document vectors (Salton 1971, p. 5) and automatic thesaurus construction by correlating cooccurrences of words in documents (Salton 1971, pp. 133-134).

In this section, we cover three categories of incrementally more complex embeddings, on the basis that it is useful to cover simpler models since some degree of intuition about the more complex models can be gained by extrapolating from them.

2.2.2.1 Word embeddings from count data and dimensionality reduction

In vector space models of word semantics, the semantics of a word are derived solely the distribution of its occurrences in different lexical contexts, equivalently its collocations with other words. Prototypically, they don't make use of any extrinsic information such word-

Table 2: Table showing a labelled latent feature matrix, F. It is one part of a factoring of Table 1.

Lat. feat.	Barks	Meows	Can stroke	Likes walks	Likes string	Lives in wild
More cat like, less dog like	-1	1	0	-1	1	0
Domesticated animal like	0	0	1	0	0	-1

Table 3: Table showing a labelled entity matrix E with the same entities as Table 1 defined in terms of the latent feature matrix given in Table 2.

Entity	More cat like, less dog like	Domesticated animal like
Cat	1	1
Dog	-1	1

definition dictionaries or dictionaries containing semantic links between words such as WordNet. The linguistic theory underlying the relation between distributional cooccurrence and word meaning is called the distributional hypothesis, summarised pithily by Firth (1957) as "You shall know a word by the company it keeps!".

We now return to the example of Table 1 to see how it can be distilled into a more compact representation using dimensionality reduction. Tables 1, 2 & 3, showing matrices D, F & E, respectively are related by a simple linear equation:

$$D = F * E$$

In general, for any D, we can always obtain some F and E by applying single value decomposition. This approach can be refined in several ways[4]

Now we have covered an illustrative example which gives some intuition of the benefits of this type of representation. However, to cover embeddings in the general case, we must relax our definition. In general, a lexical embedding is a vector space in which each lexical element is associated with a vector which somehow represents its meaning. Most typically this is derived or learnt from a corpus. The representation can be thought of as compressing information

4. See for example Jurafsky and Martin (2019b).

about many local contexts of the lexical element. In principle, it follows that some information about all of the local contexts of a lexical element could be recovered from the resulting vector.

Moving beyond the our example, we not always be able to assume that our embeddings actually contain latent features which explain our data according to our intuitions, but may obtain latent features which do explain real phenomena in the distribution underlying the sampled data but which do not have a straight forward intuitive interpretation, or we may recover features which do not represent the underlying distribution but instead noise as a result of the sampling process. Moreover, since they are obtained automatically, the features we recover are unlabelled. However, despite these caveats, we may still find that similarity measures between some pair of vectors e_i and e_j will meaningfully correspond to the similarity between the entities, which is the main property of embeddings used in this thesis.

2.2.2.2 Learnt word embeddings

Recent work in word embeddings is based on prediction rather than counting. Notable for its simplicity and efficiency is one of the two algorithms included in the word2vec (Mikolov et al. 2013) package: skip-gram with negative sampling. This technique learns two vectors for each word form: a target embedding and a context embedding. Learning takes place across many context windows, each consisting of a target word, and for example the two words to its left and right. For each (target word, context word) pair, some number of negative sample are taken. At each learning step, the target and context vectors are modified so as to move the positive context vector and target vectors closer together while moving the negative context vectors and target vector further apart.

This set up typifies the learning of word vectors. The representations were learnt from a corpus based on an objective which captures properties of a local context. To see that they really do capture these properties, we can imagine recovering local context properties of the corpus using only the embedding. With the resulting embeddings we could, for example, use a particular word's target vector to find words likely to occur its context by querying the set of context vectors for nearby vectors.

2.2.2.3 Language modelling and context sensitive word embeddings

We now turn to a family of tasks referred to as language modelling. The archetypal language modelling task is to try and predict the next word given some limited window of context. This type of task has historically been used for text generation: for example generating grammatical nonsense or for improving the fluency of Phrase Based (statistical) Machine Translation (PBMT) systems. Language modelling is a form of self-supervision. In contrast to a typical supervised setting where some input is mapped to some output, and the mapping must be reproduced, or an unsupervised setting where different instances from an input space are compared using metrics such as cosine distance, in self-supervision setting, some portion of what would typically be the input is deleted, and treated as an output to be predicted from the remaining part of the input.

Melamud, Goldberger, and Dagan (2016) pioneered the use of using gap based language modelling to obtained a useful representation for other tasks with Context2Vec. Context2Vec first makes contexts from sentences by replacing each token with a placeholder. A bidirectional Long Short-Term Memory (LSTM) Recursive Neural Network (RNN) is then trained to predict the missing word from the context. This is a neural network that processes each token twice, once from left-to-right and again from right-to-left and contains a memory like element to capture long term dependencies. The outputs from the forward and backwards pass are combined and used to predict the missing token. The resulting vector space includes both contexts and word embeddings in the same space, with words ending up near contexts in which they are likely to appear.

ELMo (Peters et al. 2018) also trains a bidirectional LSTM (Hochreiter and Schmidhuber 1997) RNN, but using a slightly different language modelling task: rather than going from contexts to deleted tokens, in both the forward pass and backwards passes the next token is predicted. The results of both passes are combined in the learning objective.

BERT (Devlin et al. 2019) learns to perform another type of language modelling task called masked language modelling. Each input sentence has some percentage of its tokens masked, and these are predicted by the model. An input can be reused many times with different masks. BERT then used a transformer model (Vaswani et al. 2017) rather than an LSTM, which is based on stacked layers of self-attention.

All of these models give us some possibility to represent either bare-contexts, (in the case of Context2Vec) or words in context (in the case of ELMo and BERT) as vectors. These contextual vectors may either be used as is, or feature selection may be used, essentially learning which parts of the vector are useful to a particular task or by fine-tuning: retraining part or whole of the neural network, or by combining these approaches. These approaches are compared by Peters, Ruder, and Smith (2019).

2.3 WSD Techniques

Some of the major approaches to WSD are laid out in Figure 3 (Jurafsky and Martin 2019a). Approaches towards the left require less structured or annotated data than those towards the right. In a Machine Learning (ML) context, it's common to talk of unsupervised and supervised techniques (Russell and Norvig 2010, p. 694-695). Supervised techniques typically have some labelled data and use this to train a model, which can label further data. Unsupervised techniques, on the other hand, associate data points together in a fixed or variable number of clusters. However, in WSD the terms are applied inconsistently, with unsupervised sometimes referring only to induction,[5] but other times referring additionally to knowledge based techniques.[6] For this reason, I will avoid referring to unsupervised WSD.

An in depth review of WSD techniques is given in Navigli (2009), while Jurafsky and Martin (2019a) provide a pedagogical overview of the area. This background section proceeds in ascending order of the amount of data required, from induction in Section 2.3.1 and knowledge based techniques in sections 2.3.2 & 2.3.3 at the low end, up to supervised techniques in Section 2.3.4 at the high end. Finally, Section 2.3.5 describes how different systems are evaluated.

5. See for example Jurafsky and Martin (2019a, Section 7).
6. See for example Navigli and Lapata (2010).

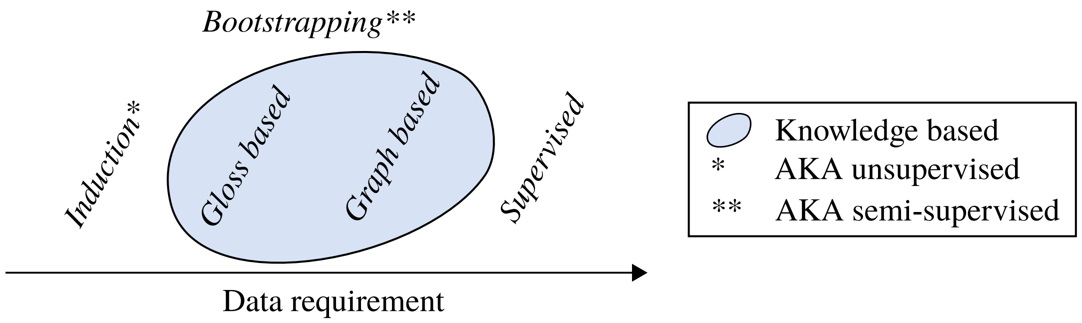


Figure 3: Diagram showing the major families of techniques for WSD. Gloss and graph based techniques are collectively known as knowledge based techniques. Bootstrapping is positioned at the same place on the x-axis as the knowledge based techniques.

2.3.1 Word Sense Induction

Word Sense Induction (WSI) techniques require the least data, abandoning even the need for a sense inventory. Rather, these techniques discover clusters of uses which share some common context. WSI is part of a larger field called linguistic structure discovery,[7] where language is analysed "from scratch", that is to say, the system is only allowed access to an unannotated corpora. A downside of this approach is that the resulting clusters are not associated with a dictionary definition, but rather only with a set of usages. However, recent work has focused on increasing the interpretability of the clusters (Panchenko et al. 2017).

Schütze (1998) produced the seminal work within this category. In this algorithm, each usage of a particular word is associated with a context vector made up of some words selected from the same paragraph or sentence. Clustering is then performed on these vectors to group usages into senses. Disambiguation can be performed by nearest neighbour classification. In this work, three main parameters were considered: which words were included in the context; whether the context vectors were subject to dimensionality reduction[8]; and the number of clusters.

Véronis (2004) addressed two shortcomings of Schütze's work. Firstly, that all words are assumed to have a fixed number of senses given as a parameter to the algorithm. Secondly, that each sense is assumed to have approximately the same frequency, which is not the case in general since many words exhibit a skewed sense distribution. For each word to be dis-

7. See for example Biemann (2011).
8. For an overview of dimensionality reduction techniques, see Maaten, Postma, and Herik (2007).

ambiguated, this technique forms a weighted graph of words and word cooccurrences within the same paragraph. A special-purpose heuristic is used to identify high density components, which are assumed to correspond to word senses.

A recent thread of work in this area is unifying WSI with the improvements to word vectors made by word2vec (Mikolov et al. 2013) and GloVe (Pennington, Socher, and Manning 2014). For example, Pelevina et al. (2016) process pre-existing word vectors to produce sense specific vectors. For each candidate word to be disambiguated, an ego-network is formed. It contains edges between the candidate word and words nearby in the original vector space, as well as second order edges between any given pair of these related words. This graph is then clustered using Chinese whispers label propagation (Biemann 2006). The final sense specific word vectors are obtained by averaging the word vectors within each cluster.

2.3.2 Gloss based WSD

The original gloss based WSD algorithm is due to Lesk (1986). This algorithm jointly assigns senses to all words in a sentence, so as to maximise the sum of the pair-wise overlapping words between their glosses. In general, calculating this without resorting to heuristics and approximations can be computationally intractable, and some subsequent approaches use a simplified version (Kilgarriff and Rosenzweig 2000b), where each word is considered individually, and a sense is chosen such that the overlap between its gloss and the other words in the sentence is maximised. Both methods can be thought of in a spatial way as working in a metric space with Jaccard (1926) distance on a Bag Of Words (BOW) representation. In this case, simplified Lesk can be described as given the BOW of a word's context, selecting the sense with nearest neighbour (Peterson 2009) based on the BOW representation of its gloss. In this setting, the original Lesk seeks a collection of sense assignments for a sentence so as to minimise the sum of the distances between all pairs of all senses' gloss BOWs. Since simplified Lesk has become the preferred approach, it is often simply referred to as Lesk.

A big problem these approaches face is data sparsity. Often a gloss produces no interesting overlap at all. These problems can be mitigated by expanding the context and/or gloss, or by representing words by dense word vectors (Jurafsky and Martin 2019b). An example of the first approach is Banerjee and Ted Pedersen (2003) expand the gloss of each term with the glosses of related senses according to a computer thesaurus. Recent work takes the second approach using dense word vectors which causes similar words to end up near to each other in a vector space. In Basile, Caputo, and Semeraro (2014) each word is represented by a word vector, and the words in the context and gloss are each averaged into two vectors. This type of representation is sometimes referred to as Average/Aggregate of Word Embeddings (AWE). WSD then operates as usual, by choosing the nearest neighbour.

2.3.3 Graph based WSD

Graph based WSD algorithms disambiguate by using a computer thesaurus, a type of LKB which includes links between word senses, such as synonymy, hypernymy (an is-a relationship) and metronymy (a part-whole relationship) forming a graph. WordNet (Miller 1995), and more recently BabelNet (Navigli and Ponzetto 2012), which is a multilingual superset of WordNet are commonly used LKBs for performing this type of WSD. An early piece of work in this area was conducted by Quillian (1969), who used the path length in a graph of this variety as a measure of semantic distance. The idea here is that is to say word senses which are close in the graph are considered related, a key idea underlying graph based WSD algorithms.

There are multiple ways to use this intuition for WSD. Navigli and Lapata (2010) present a systematic comparison where two varieties of graph based WSD are considered. In the variety based on a local connectivity measure, a graph is formed by considering the subgraph of the LKB containing all senses of all words in a sentence. Next, for each word, the sense with maximum local connectivity is chosen. Measures tested include a sense node's out-degree, that is the number of edges leaving a node, and PageRank (Page et al. 1999), the measure originally used by Google to determine a web page's important in a link graph. In the variety based on a global connectivity measure, a graph is made for each possible combination of word senses in a sentence, and the sense combination with the highest measure of the overall graph connectivity is chosen. Note that this may be become intractable without resorting to approximations or heuristics. An example of this type of measure is edge density, which is the fraction of edges in the graph over the number of edges in a complete (full connected) graph.

Moro, Raganato, and Navigli (2014) first preprocess the LKB to produce a semantic signature for each node by performing random walk with restart and including nodes visited over a threshold number of times. Because this algorithm also performs named entity tagging, a situation where multiple surface forms can refer to a single sense e.g., “IBM” and “Big Blue”, another graph is constructed with all surface form, sense combinations as nodes, linked edges according to each sense’s semantic signature. The rest of the algorithm proceeds as a combination of the local and global varieties of graph based WSD algorithms. The graph is pruned to find a subgraph with high global connectivity according to the measure of average degree. Nodes with the highest local connectivity score, which includes degree as a factor, are then chosen.

2.3.4 Supervised

The typical approach for applying supervised WSD is to train individual models for each word. Each model knows only have to disambiguate this single word. This approach is sometimes referred to as the word experts approach, as a reference to older systems where systems of rules (expert systems) were written for each word to determine a sense.

The major caveat with this approach is we must face Heaps’ law at the word sense level (Heaps 1978, p. 208, sec. 7.5). Heaps’ law states that the size of the vocabulary grows in a concave power law with the number of tokens in the corpus. Restated, increasing the size of our corpus has diminishing returns in terms of increasing the coverage of the vocabulary. This law is a consequence of the skewed distribution of words. However, word senses also tend to exhibit heavily skewed distribution. Thus, even for very common polysemous words, it may become difficult to get a good coverage of all senses. This situation has also been referred to as the data-acquisition bottleneck.

As mentioned in Section 2.3.2, Simplified Lesk is, in fact, a nearest neighbour classifier, but using data from the word definitions rather than labelled examples. Analogous to this we can perform nearest neighbour classification with labelled examples. The analogy of Simplified

Lesk would then be a BOW classifier based on Jacquard distance. The next step would be an AWE and cosine distance based classifier. Beyond this, there is the possibility of taking account of whole contexts, as with *Context2Vec*, summarised in Section 2.2.2.3. Melamud, Goldberger, and Dagan (2016) found that using a nearest neighbour classifier with their context embeddings outperforms AWE to the point where it reaches comparable performance with the state of the art WSD approaches they compare with.

Zhong and Ng (2010) introduce the popular WSD program *ItMakesSense* based on Support Vector Machines (SVMs). ItMakeSense which has subsequently both become a baseline for supervised WSD and also been extended due to its modular architecture. Specifically different selections of features, which together make up the vector fed to the classifier can be chosen. One type of feature there has been recent interest in is word embeddings. Iacobacci, Pilehvar, and Navigli (2016) evaluated the effect of adding different word embedding features to *ItMakesSense*.

2.3.5 Evaluation

For WSD, the measures of precision, recall and F_1-measure are typically used (Manning, Raghavan, Schütze, et al. 2008, sec 8.3). Consider a label produced by the system which agrees with the ground truth to be a true positive, and denote the number of occurrences thereof as tp. Consider also a label produced by the system but not in the ground truth as a false positive, with count fp, and vice versa a label in the ground truth not produced by the system as a false negative, with count fn. Precision and recall can now be defined as:

$$precision = \frac{tp}{tp + fp} \qquad recall = \frac{tp}{tp + fn}$$

F_1-measure is then the harmonic mean of these two measures:

$$F_1 = \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

One of the most commonly used metrics in classification is accuracy:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

The F_1-measure was developed for evaluating systems in the field of information retrieval. The reason accuracy was not used in this context is that in a typical case there will be a large number of false positives: irrelevant documents which are not retrieved. This creates an unbalanced situation in which a system which returns no documents would get a quite high accuracy due to the tn term in the numerator. A similar situation is in play with WSD, where there is typically a large number of irrelevant senses that can be assigned to each token, hence F_1-measure is used.

Due to a lack of availability of gold standard corpora to compare against, early evaluation of WSD systems consisted mainly of qualitative examples of how the system performed on a few, hopefully indicative, examples. In the first quantitative evaluations, a few ambiguous words were chosen, and a human annotator would disambiguate only these words in a corpus so that the accuracy of the system could be measured. Gale, Church, and Yarowsky (1992) note the drawbacks of this approach: “This technique is not without its problems, perhaps the worst of which is that the sample may not be very representative of the general vocabulary”.

Gale, Church, and Yarowsky (1992) also note the difficulties of comparing results with previous work when there are so many potential confounding variables to control: “One feels uncomfortable about comparing results across experiments, since there are many potentially important differences including different corpora, different words, different judges, differences in treatment of precision and recall, and differences in the use of tools such as parsers and POS taggers, etc.”. In response to these problems, the academic community began to organise a series of shared evaluations for WSD. The first such workshop, SENSEVAL, was organised in 1998 (Kilgarriff and Palmer 2000). In the shared evaluation model, research groups or other interested parties evaluate their WSD systems against the same *ground truth*, a corpus where all words have been manually labelled with the correct sense. Since the evaluation step is shared, confounding variables can be controlled, and more effort can be invested in producing a high quality ground truth. After the evaluation, the results and descriptions of competing systems are disseminated.

As well as comparing the systems against each other, they are also compared against a series of baselines. Many of the knowledge based techniques have difficulty beating a simple baseline that assigns each word its Most Frequent Sense (MFS). For example, in SemEval[9] 2015 task 13 (Moro and Navigli 2015) competing systems had to disambiguate a multilingual corpus using the BabelNet sense inventory. For English, no system managed to beat the MFS baseline. For Spanish and Italian, however, a PageRank based local connectivity measure graph based WSD system performed best in terms of F_1-measure (Manion 2015).

Raganato, Camacho-Collados, and Navigli (2017) recently performed an independent systematic comparison where they harmonised evaluation data from previous SENSEVAL and SemEval WSD tasks into a common format. This included, for example, updating them all to use the newest version of WordNet as a sense inventory. This allowed fair comparisons both between systems and different evaluation ground truths. For all previous SENSEVAL and SemEval evaluation ground truths, among the knowledge based systems evaluated, one of three systems performed best: the dense vector gloss based system of Basile, Caputo, and Semeraro (2014); the Babelfy graph based system of Moro, Raganato, and Navigli (2014); or the MFS baseline.

2.4 SLA and CALL

In SLA, the seminal work of S. D. Krashen (1982) puts forward five hypotheses about how language is acquired. Most famous is the input hypothesis, which states that language is acquired when learners comprehend input just above their current level. This is also referred to as the "i+1" hypothesis, where the learner follows an inductive process of acquiring "i+1" comprehensible input. The acquisition-learning distinction and monitor hypotheses essentially state that language acquisition is an unconscious act separate from conscious learning. The former occurs *only* through the inductive process comprehending i+1 input. The later is useful in so far as it helps the former, mainly via self-correction, but can also be harm-

9. Starting with SemEval-2007, the SENSEVAL workshops were renamed to SemEval to reflect a broadening of the scope.

ful, causing overcorrection. The hypotheses are the theoretical basis of Krashen's natural approach, (Krashen and Terrell 1998) which includes the suggestion that authentic texts of interest to the learner should be used for language learning.

More recent theories of SLA, have provided some evidence that input alone is not always sufficient for older language learners to gain high precision of all grammatical features (Lightbown and Spada 2013, pp. 168–175). Key to this is the idea that attention is required for the new linguistic knowledge to enter the memory, formulated in the noticing hypothesis by Schmidt (1990)[10]. The noticing hypothesis does not state that learners should be given additional rules (which according to Krashen are only useful to the monitor). Rather, that if a learner does not pay attention to certain features in a text, they may not acquire them. One possible interpretation is that older language learners tend to have more selective attention than younger learners.

The noticing hypothesis can be interpreted on both the micro level, for example noticing which inflection a word is in in a particular context, and on the macro level, for example, paying close attention to a text at all. The former suggests that when certain features have not been acquired, we may need to bring them to the learner's attention. The latter ties into a more general point that learner's input should be something they think is worthy of their attention, for example, a text relevant to one of their interest, or material they have to read to complete a task, such as reading tax regulations for completing a tax return. S. Krashen (1989) refer to reading a wide variety of texts in the L2 language as extensive reading, but this method has also been referred to as free reading to emphasise the aspect of learner freedom.

In an attempt to apply this, Sharwood Smith (1993) suggests input enhancement. It is suggested that instructors should modify natural texts to draw attention to salient features such as POSs or certain grammatical morphemes. If we wish to allow learners to direct their own choice of texts, however, this input enhancement should be automatically generated as and when it is needed by computer software.

This leads us to the related but distinct notion of electronic reading assistants. In contrast to input enhancement, reading assistants are focussed on helping readers obtain the meaning of

10. This theory at least partially contracts Krashen's ideas, and has a fair share of vocal detractors, including Truscott (1998).

the text rather than helping them acquire particular largely grammatical knowledge which may not be strictly necessary for understanding. If such systems ultimately assist comprehension, they could potentially transform i+2 input into i+1 input.

Another related notation is that of Data-Driven Language Learning (DLL), a term coined by Tim Johns. The modern process for authoring dictionaries and language learning materials makes extensive use of data derived from corpora. This includes word frequency data as well as concordances: long lists of words surrounded by context words. DDL "attempt[s] to cut out the middleman as far as possible and to give the learner direct access to the data" (Boulton 2009, quoting Tim Johns).

Meurers et al. (2010) suggests that software which produces enhanced input should be called ATICALL (Authentic Text Intelligent Computer Aided Language Learning) systems. Related to this are reading support tools, which offer access to some reference resource such as a dictionary, alongside the text. The rest of this section reviews a few of these systems. Some of these systems contain other CALL elements such as exercise generation, but this is not covered here.

2.4.1 Concordance and DDL based approaches

The Sketch Engine (Kilgarriff et al. 2014) is probably the most prominent commercial system of this type. While the main product is primarily marketed at lexicographers and research linguists, Sketch Engine for Language Learners (SKELL)[11] (Kilgarriff et al. 2015) is targeted directly towards learners, in the tradition of DLL.

The Flax interactive language learning system[12] (Fitzgerald, Wu, and Marín 2015) can display collocations based on different collections of corpora. The collections include those chosen by the authors, like the British National Corpus, as a section where users can upload new content. The system can also automatically extract synonyms based on collocations.

There are various approaches to extracting collocations from corpora. Most systems try to distinguish between different types of constructions e.g., when looking at collocations of

11. Accessible at `https://skell.sketchengine.co.uk/`.
12. Accessible at `http://flax.nzdl.org/greenstone3/flax`.

"say" we might like to distinguish between subjects e.g., "officials" and objects e.g., "nothing". Bhalla and Klimcikova (2019) gives an overview of the different approaches, which include regular expression style pattern matching on POS tagged text and on parse trees.

Collocations which are particularly difficult for language learners can be extracted by referring to parallel corpora. Moirón and Tiedemann (2006) extracted idioms, that is groups of words with non-compositional meanings, by extracting candidates of three words using patterns in dependency trees and measuring when they are aligned to a single word with high translation entropy, indicating that one word maps more or less uniformly to multiple. Graën and Schneider (2017) predicted the most likely errors a native L1 learner of L2 will make for particular verb preposition combinations (e.g., "suffer from") by using frequency information on a word aligned corpus to rank the possible prepositions after translation from L2 to L1 and back to L2.

2.4.2 Enhanced input approaches

As well as showing concordances, Flax (Fitzgerald, Wu, and Marín 2015) also provides tools for enhanced input. It can highlight particular POSs, groups of words by frequency and topic specific words.

Zilio, Wilkens, and Fairon (2017) introduce the Smart and Intelligent Language Learning Environment (SMILLE)[13]. SMILLE works by utilising a web crawler and presenting a modified version of the fetched web page to the language learner. It can highlight different grammatical structures, grouped by their Common European Framework of Reference for Languages (CEFR) level.

WERTi[14] (Working with English Real Texts interactively) (Meurers et al. 2010) has a similar architecture in that it relies on crawling web pages. It can highlight POSs as well as different grammatical structures. It can also display several annotations simultaneously. VIEW[15] (Vi-

13. This system is not currently publicly accessible, It is being developed commercially with Altissia and is indented to be made available to Erasmus+ exchange students at `https://erasmusplusols.eu/`.

14. Source code available at `https://github.com/adimit/werti`.

15. Available at `http://sifnos.sfs.uni-tuebingen.de/VIEW/`.

sual Input Enhancement for the Web) Reynolds, Schaf, and Meurers (2014) is the successor of WERTi, and is distributed as a browser extension.

2.4.3 Reading assistants

SMILLE (Zilio, Wilkens, and Fairon 2017) also acts as a reading assistant. It can provide word definitions based on a user click from the freely available WordNet and the commercial Merriam-Webster dictionaries. It can also show grammar reference information about the user selected area when a construction from its database is recognised.

Nerbonne, Dokter, and Smit (1998) presents GLOSSER, a system for Dutch learners of French. Both online[16] and desktop versions of this program were developed, but only the desktop version could work with reader selected texts. The authors noted the potential of WSD for this type of system and implemented it such that words in manually chosen, unambiguous fixed phrases such as guerre mondiale “world war” are given the appropriate word sense (this is a low recall, rule based WSD system).

16. Accessible at `http://www.let.rug.nl/glosser/Glosser/`.

3 Automatically constructing a sense tagged corpus

Both the training of supervised Word Sense Disambiguation (WSD) algorithms and the evaluation of any WSD algorithm requires a sense annotated corpus. The manual construction of sense annotated corpora requires time and language specific expertise. This is sometimes referred to as the knowledge acquisition bottleneck.

To avoid this manual work, some researchers have attempted to automatically generate sense tagged corpora by reusing existing knowledge in different ways. Pasini and Camacho-Collados (2018) give a recent overview of sense tagged corpora including ones generated using these techniques. The existing knowledge can take the form of apriori knowledge about a text, such as knowing that word definitions are likely to refer to the same sense that they are describing or that a Wiktionary link refers to the sense it links to.

Another approach is to take advantage of the differences in word senses between different languages. An ambiguous word in one language may be unambiguous in another. Given two ambiguous words in two languages referring to the same thing, it may be that there is only one possible unambiguous interpretation, or at least that there is less ambiguity left after considering both languages together. Taghipour and Ng (2015) introduce OMSTI, a corpus based on a technique introduced by Chan and Ng (2005) where English words are tagged with a WordNet sense based on word aligned Mandarin Chinese words.

Mandarin Chinese is used as a disambiguating language for English because it is assumed that corresponding English and Mandarin words have different etymologies and so may have picked up different sets of senses, resulting in the confusion classs[1] of two tokens referring to a concept being small (ideally just one sense).

Other techniques use knowledge based WSD to generate a sense tagged corpus. Knowledge based WSD can perform poorly in some cases, so to make this work, something else needs to be added. Usually, the best algorithms provide reasonably reliable confidence scores, so to improve the precision of the resulting corpus filtering can be applied to only keep those annotations with some minimum confidence score (Pasini and Navigli 2017).

1. In this case the confusion class is the set of word senses a token can have.

Finally, EuroSense (Bovi et al. 2017) is a multilingual sense tagged corpus. It is constructed using the EuroParl multilingual corpus. Words in European languages are more likely to share a confusion class, since many of them have the same etymology, and thus this may seem to be poor material for disambiguation. EuroSense sidesteps this issue in two ways. First, it considers n-way parallel sentences rather than just sentence pairs, meaning if just one of the n languages has a different confusion class, we can disambiguate. Secondly, the actual disambiguation is based upon the Babelfy (Moro, Raganato, and Navigli 2014) graph based WSD algorithm, which can take account of all words in the sentence. An extensive review revealed this as the only publicly available sense tagged corpus of Finnish.

As part of this thesis, a second sense tagged corpus is constructed using a method similar to OMSTI. The corpus is induced from Mandarin-Finnish parallel texts. It is argued here that while this is not necessarily the best possible pair to choose, it is a reasonably good pair which trades off the following factors:

- The level of mutual disambiguation the pair provides.
- The amount of parallel text available for the pair.
- The availability of language resources for the second language. The language must have at least one WordNet with reasonable coverage.

Similar to Mandarin-English, this pair is hoped to provide a high level of mutual disambiguation. Knowing that Finnish is part of the Uralic language group, distinct from the Indo-European language group which contains most other European languages, the question might arise: why not use a European language? For example, why not use English-Finnish, since this pair provides a very well resourced second language where a lot of parallel text is available. The answer is that it is unlikely to provide the level of mutual disambiguation which may be assumed based upon the difference in language families.

Häkkinen (1992) looks at the etymology of Finnish words by selecting 1888 core common lemmas and reducing them to 844 root stems by discarding words resulting from any form of morphological derivation. Of the resulting lemmas, 49% are determined to originate from Indo-European languages, while 46% appear to be in some sense original to Proto-Finnic. Kallio (2012) notes that contact with Germanic and Proto-Germanic may stretch back as far as the Neolithic period. Importantly Finnish has had multiple periods of contact with languages with higher prestige for almost a millennium: Swedish as a result of Swedish rule and the resulting Swedish speaking merchant class; German as a result of the Hanseatic league; Russian during Finland's time as a Grand Duchy under Russian rule; and then in the 21st century English as a result of the post-war Anglo-American hegemony. Contact with prestige languages like this is known to accelerate the process of language mixing and synthesis[2].

All this noted, we do not necessarily care about the source of the individual words in the lexicon, but rather the degree of mutual disambiguation two tokens referring to the same concept might provide. These notions are somewhat related but differ since the senses which words refer to can evolve. One process through which word senses evolve is when new senses are created by a sense extension process, using metaphor or analogy[3]. Here, the view is taken that sense extension is essentially analogous to word derivation, e.g., by compounding or morphological derivation. It is known that a word derivation process can transfer across languages. An example of such a structural loan, or calque, is the Finnish "voileipä|kakku" which is loaned from the Swedish "smörgås|tårta" (English: "sandwich cake"). It is assumed by analogy then, that sense extensions can also travel across languages in close contact — just the type of contact Finnish has had with other European languages. Thus, due to close contact over a long period, a single European language may not provide a high level of disambiguation for Finnish. In addition, a sense tagged corpus based on one of the largest and most popular sources of parallel data of the European languages, EuroParl, already exists in the form of EuroSense, removing an attractive potential source of parallel text.

Mandarin has also been in contact with the European languages since at least the 13th century[4]. However, this language contact has occurred over a significantly longer distance, and as we go back through history, earlier language contact is increasingly limited by geography. Thus it is hoped that many Mandarin words retain reasonably distinct confusion classs

2. For a general account, see for example Thomason (2001). For a specific account of structural changes resulting from Finnish-Swedish contact, see Häkkinen (1997) and De Smit (2006).

3. Different metaphorical processes are given and explored in a general context in Sweetser (1990). Srinivasan and Rabagliati (2015) give a recent empirical work comparing a battery of metaphorical processes across languages.

4. For an overview see Vervaet (2017, Chapter 2).

with respect to Finnish. Additionally, Mandarin is a well resourced language with multiple WordNets, introduced in Section 3.1.

Parallel corpora were obtained from the Open Parallel CorpUS (OPUS) (Tiedemann 2012). OPUS contains many subcorpora. Some of the most extensive, highest quality parallel corpora which are freely available are produced by those international organisations which routinely operate in a highly multilingual environment: the European Union (EU) and the United Nations (UN). Unfortunately, Mandarin is not an official EU language, and, as noted, a sense tagged Finnish corpus based on EuroParl is already available. Since Finnish is not an official UN language, this also rules out parallel texts from the UN. The largest subcorpus of OPUS containing Mandarin and Finnish is the OpenSubtitles2018 corpus (Lison, Tiedemann, and Kouylekov 2018), which is composed of subtitles from films and television programmes. The rest of this chapter outlines the creation of the Sense Tagged Instances For Finnish (STIFF)[5] corpus from the raw material of the OpenSubtitles2018 corpus.

3.1 Preprocessing and sense inventory of Mandarin Chinese

Chinese, or Hànyǔ, is a language group[6] spoken predominantly in East Asia. Many of the languages within the group never developed a standardised written form; they lack the status of being an official language, as well as prestige within mainstream culture. Mandarin Chinese, on the other hand, has not one, but two official writing systems stemming from two character sets. Simplified Chinese characters are officially used in mainland China and Singapore, while traditional Chinese characters are officially used in Taiwan, Hong Kong and Macau. In practice, there are also small differences between the writing systems in Hong Kong and Taiwan e.g., the preferred form of the Pinyin romanisation lǐ, meaning “inside” or “within”, is 裡 in Taiwan and 裏 in Hong Kong.

5. The source code has been made available at `https://github.com/frankier/STIFF`.

6. Although its constituent languages are sometimes referred to as dialects of a single language, this is not the linguistic perspective. DeFrancis (1984) writes “To call Chinese a single language composed of dialects with varying degrees of difference is to mislead by minimizing disparities that according to Chao are as great as those between English and Dutch. To call Chinese a family of languages is to suggest extralinguistic differences that in fact do not exist and to overlook the unique linguistic situation that exists in China.“

Here, only the simplified and traditional forms of standard Mandarin Chinese are used. At first blush, mapping between traditional and simplified characters might be assumed to be a matter of performing a simple character-by-character operation. Halpern and Kerman (1999) point out some pitfalls of this approach. Neither the character-by-character traditional to simplified (hereafter `t2s`) conversion nor the simplified to traditional mapping (hereafter `s2t`) are completely functional/many-to-one. For example, in `s2t` 发 can map to either 發 or 髮[7], and in `t2s` 徵 can map to either 徵 or 征. Moreover, neither mapping, when performed character-wise, is fully idempotent. 薴, when the `t2s` mapping is applied once, will map to 苧; when applied again, it will map to 苎. However, this triple of characters is the only case of this.

3.1.1 Mandarin in OpenSubtitles2018

Each film subtitled in Finnish and Mandarin in OpenSubtitles2018 forms a bilingual text or bitext. There are two Mandarin language codes in OpenSubtitles2018: zh_CN and zh_TW, referring to Mandarin as used in mainland China, and Taiwan respectively, i.e., Mandarin Chinese written with simple and traditional characters respectively. Figure 4 shows how many bitexts are tagged with each language code.

How accurate are these labels? To verify this a simple detection scheme was created. The detection scheme was built on top of OpenCC (Open Chinese Converter)[8], which performs Chinese character set conversion. The detection scheme proceeds by converting each sentence using both the `t2s` and `s2t` converters from OpenCC, then classifying according to Table 4. To determine whether a film subtitle was written with a particular character set, character set detection was run sentence-wise. If the fraction of sentences detected as either traditional or ambiguous is above $\frac{2}{3}$, the subtitle is classified as traditional. Correspondingly, if the fraction of sentences detected as either simplified or ambiguous is above $\frac{2}{3}$, the subtitle is classified as simplified. A film meeting neither or both of these criteria is classified as neither.

7. This and the following examples are derived from data in the Unicode Han Database. See `https://www.unicode.org/reports/tr38/`.

8. Obtained from `https://github.com/BYVoid/OpenCC`.

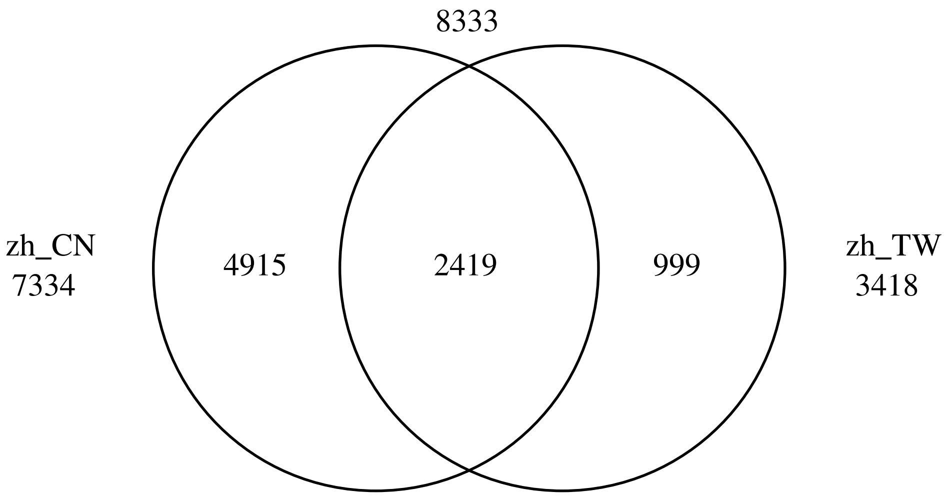


Figure 4: A Venn diagram showing the number of Finnish-Mandarin bitexts (films) tagged as zh_TW and zh_CN within the OpenSubtitles2018 corpus.

Table 4: Chinese character set detection scheme.

sentence = `s2t`	sentence = `t2s`	detected as
✓	✓	ambiguous
✓	✗	traditional
✗	✓	simplified
✗	✗	neither

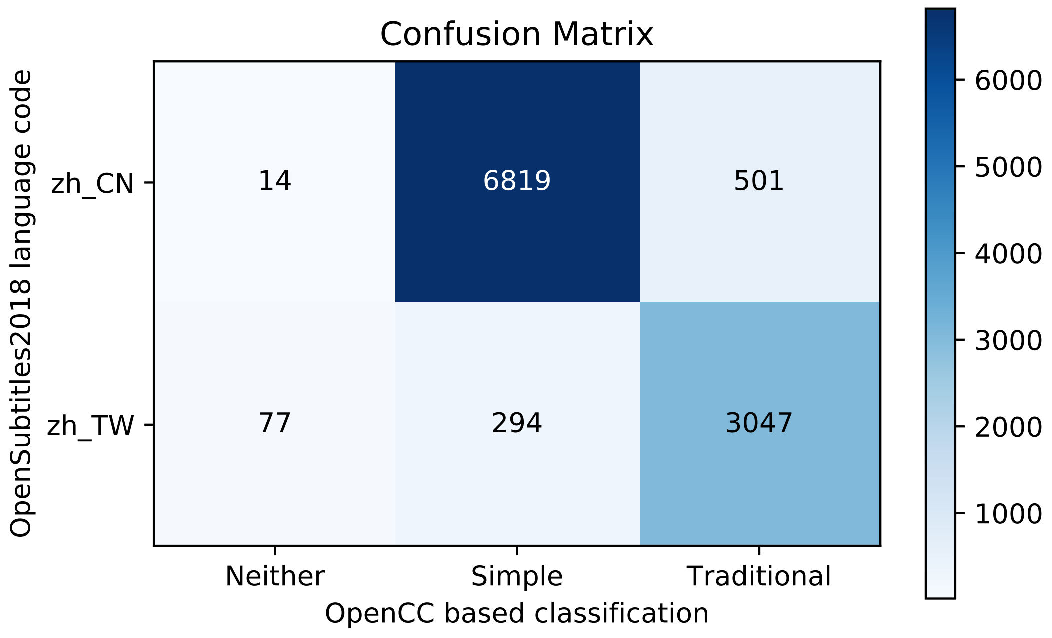


Figure 5: Confusion matrix showing character set classification of films in OpenSubtitles2018 against their reclassification with the OpenCC based scheme.

A confusion matrix showing the result of this character set detection is shown in Figure 5. Manual inspection of a few subtitles classified as neither reveals that the phenomenon is at least partially due to encoding problems: in particular, it appears that in some cases more than one text encoding has been used within a particular file resulting in a failure to fully convert the file to UTF-8. Assuming it is possible to recover usable text, doing so would require going back to the original source subtitle files used to create OpenSubtitles2018, which are not readily available, and fixing the OPUS processing pipeline[9]. However, since it only affects a small part of the corpus ($\sim$1%) overall doing so is not essential to the task at hand.

3.1.2 Mandarin WordNets

There are two freely available manually created Mandarin WordNets. The Chinese Open WordNet (Wang and Bond 2013) was produced in Singapore and is based on the simplified

9. Which can be obtained from `http://opus.nlpl.eu/tools.php`.

Table 5: Table showing the pseudo language code, as well as the number of synsets, words, senses, and single character words of the considered Mandarin WordNets. Partially reproduced from `http://compling.hss.ntu.edu.sg/omw/`.

WordNet	Code	Synsets	Words	Senses	Chars
Chinese Open WordNet	`cmn`	42,312	61,533	79,809	1297
Chinese WordNet (Taiwan)	`qcn`	4913	3206	8069	965
Extended Chinese WordNet	`qwc`	12,130	16,171	19,079	1236

character set, while the Chinese WordNet (C.-R. Huang et al. 2010) was produced in Taiwan and is based on the traditional character set. Additionally, Bond and Foster (2013) created another simplified character set Mandarin WordNet automatically based on data obtained from Wiktionary.

Some information about the size of the WordNets, as well as the codes they are referred to by subsequently, are given in Table 5. As can be seen, `cmn` is by far the largest WordNet. However, in terms of coverage of single character lemmas, which occur more frequently than multi character lemmas, (note that every multi character lemma contains single character lemmas) it is of a similar size to the others. All WordNets are considered collectively so as to give good coverage of single character lemmas.

3.1.3 Obtaining Mandarin lemmas

As we have seen in sections 3.1.1 and 3.1.2, considering both traditional and simplified characters together increases the size of the corpus and lexicon respectively as well as side stepping any potential problems caused by incorrectly classifying the character set of some text. Thus, character conversion is used. All text, including sentences from subtitles and lemmas from WordNet, regardless of its original script, is converted `t2s` character by character. As noted previously, this is not entirely without pitfalls, (it is a one-to-many, non-idempotent mapping) but is a simple scheme which works sufficiently well in this case. In particular, one failure case is that a character which is in traditional is mapped to the wrong simplified character and therefore cannot be matched with the character it should have been mapped to. This is a consequence of the one-to-many nature of the mapping. However, there are only three characters that are one-to-many, 鍾, 顧 & 餘. Another failure case is that a traditional character could be mapped to a simplified character, but it cannot be matched with the simplified

Table 6: Example showing how substring search on unsegmented Mandarin text can produce extra lemmas: both True Positives (TPs) and False Positives (FPs).

Type	Text	Extracted
Segmented	到处 都 是 幽灵	到处: everywhere, 都: all, 是: to be, 幽灵: ghosts
Unsegmented	到处都是幽灵	+ 到处都是: to swarm ✓
Segmented	修复 工作	修复: repair, 工作: task
Unsegmented	修复工作	+ 复工: rework ✗

character it was mapped to because this character is simplified again to a different simplified character. This is a consequence of the non-idempotent nature of the mapping, but as noted there is only 1 idempotent character. Given the low supposed frequency of these cases, they are not dealt with specially.

Words in written Mandarin text are not delimited by any space. As such, the first stage of many Mandarin text processing systems is word segmentation: splitting sentences into words. OpenSubtitles2018 contains both segmented and unsegmented versions of the Mandarin text. To obtain Mandarin lemmas, both are used.

The segmented text is used because word alignments based upon the segmentation are used in later steps. For single words, extraction can proceed by looking up each word directly. However, `cmn` also contains Multi-Word Expressions (MWEs), separated by a +. These are both looked up as single words, by removing the +, and also tried as MWEs in a similar way to Finnish as described in Section 3.2.

To obtain lemmas from the unsegmented text, they are extracted according to substring search. This can produce erroneous words formed from the end of one true word with the beginning of another, however, the aim is to generate everything possible and then eliminate low quality annotations later on in the pipeline. That is to say, the extraction step is high recall/low precision. This type of extraction allows lemmas to be extracted even if a segmentation error has been made in the segmented text. A library implementing an Aho-Corasick automaton (Aho and Corasick 1975) is used for efficient substring search [10]. Table 6 shows how substring search can produce extra lemmas, both TPs and FPs.

10. Obtained from `https://github.com/WojciechMula/pyahocorasick`.

3.2 Preprocessing and sense inventory of Finnish

Finnish is a language with rich morphology. Thus, to find potential word senses from a text, at very least lemmatisation should be performed. Omorfi (Pirinen 2015b) is a freely available morphological analyser for Finnish, which can be used for lemmatisation. Omorfi is a high recall system, including even some rarely used forms. For systems which need higher precision output, on possibility is to combine it with a morphological disambiguator/Part Of Speech (POS) tagger. FinnPOS (Silfverberg et al. 2016) is one such system which also includes a guesser to guess out of vocabulary lemmas.

The design of STIFF is to generate everything possible as the first step, including all relevant lemmas. This should include lemmas removing derivational morphemes as far as possible[11]. Sometimes, Omorfi will not remove all levels of derivational morphemes. To increase recall, each time lemmas are extracted from an Omorfi analysis, they are analysed again, until the set of found lemmas reaches a fixed point. Ultimately, all of the lemmas from this process, the lemma from FinnPOS and the original surface form are used to look up word senses from FinnWordNet (FiWN).

FiWN also contains MWEs. Here a shallow method is used, which can only extract MWEs which are contiguous in running text. However, apart from this caveat, it is high recall. We consider both the tokens in one of FiWN's MWEs and the tokens in the text under analysis to be any of the lemmas which result from analysing it. This naturally leads to a confusion network structure, illustrated in Figure 6. We now seek all paths that span a FiWN MWE and also occur anywhere in the confusion network corresponding to the analysed text. To achieve this, the FiWN MWE confusion network is made into an Aho-Corasick automaton (Aho and Corasick 1975), similar to Section 3.1. The library used[12] was modified to so that it could be used with edges at the token level rather than the character level.

Multiple pointers are kept into the MWE automaton, and every possible transition which matches a transition out of the current node in the confusion network corresponding to the analysed text is taken. This can result in redundant pointers into the Aho-Corasick automaton.

11. Currently, this is not always possible since Omorfi's derivational morphology is less developed relative to its other features.

12. Obtained from `https://github.com/WojciechMula/pyahocorasick`.

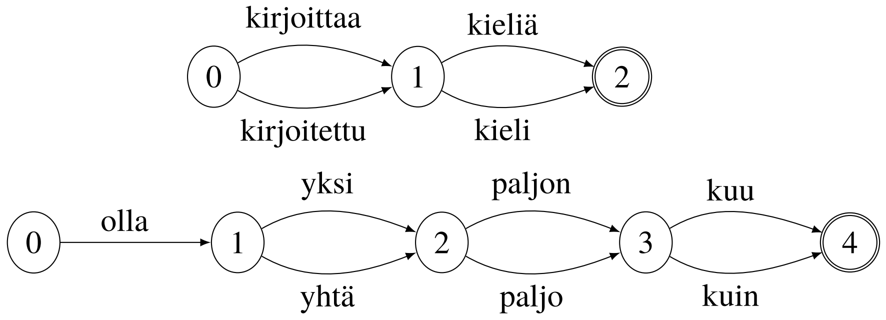


Figure 6: Confusion network arising from the Finnish MWEs "kirjoitettu kieli" and "olla yhtä paljon kuin".

The key property of an Aho-Corasick automaton is that it can continue accepting any string which begins at any of its previous states. In practice, this that it a particular node it will accept any string beginning with any suffix of the path between the current node and the start node. This leads to a notion of domination which can be used to eliminate redundant pointers. Given some pointer which has some route r_1 from the start node, it is dominated by a pointer with route r_2 from the start node if r_1 is a suffix of r_2. Here, this property is only used to remove the root node when some other node is in the current set of pointers.

3.2.1 The problem of missing senses in FinnWordNet

FiWN follows the structure of WordNet, and thus has some notable word sense omissions. WordNet focusses on words that clearly denote one or more concepts, however, there are many words that play a more grammatical role. The most reliable way of distinguishing between words with a more or less grammatical role is whether they are part of a closed class of words or not. A closed class of words is one where it is not usually possible to coin a new word within the class, e.g., the class of articles or the class of prepositions in English.

When a word is ambiguous between a sense in WordNet and a sense not in WordNet, it may not be possible to assign the word the correct sense. Many WSD algorithms assume that if a token has a set of senses available in the Lexical Knowledge Base (LKB), it must belong to one of them; It is not possible to skip tagging a token if it belongs to none of the senses. This is an example of closed world reasoning and ultimately means that it is not possible to recover from missing information in the LKB. In English, this is not a significant problem in practice since the surface forms missing from two major categories omitted from WordNet, pronouns and modal verbs, such as *she* and *must* respectively, do not usually have another sense in WordNet. For Finnish, however, we are left in an undesirable situation for multiple surface forms. *Pitää* can be used in certain constructions as a modal verb[13] as in 'mun pitää tiskata' (I **must** do the dishes) but has additional, non-modal senses — which are in FiWN — such as in 'mä pidän tiskaamisesta' (I **like** doing the dishes). Ambiguities resulting from ambiguous surface forms can also occur. *Siitä* can either be the quite common elative form of 'se' (it) or the infinitive form of 'siitä', a quite uncommon verb which nonetheless has a sense in FiWN where 'se' does not. To overcome this, it is necessary to determine whether a word is in a closed class, and if it is, remove it from consideration.

3.2.2 Estimating sense frequency of Finnish lemmas based on English data

Many techniques based on WordNet assume it is possible to pick the most likely sense of a particular set of word senses by picking the "first sense". The reason this works with Princeton WordNet is that word senses are numbered based on descending order of sense frequency based on data in SemCor (George A Miller et al. 1993), a sense tagged corpus based on the Brown corpus. Specific examples of usages of this technique are in OMSTI, where it is used to remove any ambiguity left after mutual disambiguation and as in the "first sense" baseline - a WSD algorithm which works by just picking the first sense.

Since this data is potentially needed even before a sense tagged corpus is available, it is estimated based on the frequency data in Princeton WordNet (PWN). FiWN is aligned with PWN at the lemma level. Unlike most PWN aligned WordNets, which are aligned at the synset level, FinnWordNet is aligned with PWN at the lemma level. An example of when this distinction takes effect is when lemmas are structurally similar. For example, in the synset "singer, vocalist, vocalizer, vocaliser", the Finnish lemma laulaja is mapped only to singer rather than to every lemma in the synset. When there is no clear distinction to be made, whole synsets are mapped. This reasoning fits with the existing structure of PWN: Relations between synsets encode purely semantic concerns, whereas relations between lemmas encode so called morpho-semantic relationships, such as morphological derivation.

13. For a full account of modal verbs in Finnish, see Hansen and De Haan (2009).

Let the Finnish-English lemma mapping be denoted $\mathcal{L}$, the specific frequency estimate for a Finnish lemma is then defined like so:

$$\text{freq}(l_{fin}) = \sum_{(l_{fin}, l_{eng}) \in \mathcal{L}} \frac{\text{freq}(l_{eng})}{|\{(l_{other-fin}, l_{eng}) \in \mathcal{L}\}|}$$

The rationale being that this causes the frequencies of English lemmas to be evenly distributed across all the Finnish lemmas which they map to.

To integrate the resulting synthetic frequency data into as many applications as possible, it is made available in the WordNet format[14]. The WordNet format requires sense occurrence counts, meaning the frequency data must be converted to integer values. To perform this conversion all frequencies are multiplied by the lowest common multiple of the divisors in the above formula. Some care must be taken in downstream applications since the resulting counts are no longer true counts, but rescaled probabilities. For example, if some application uses +1 smoothing, it should be rescaled to a larger constant, in this case +1000 is reasonable given the magnitude of the numbers involved.

3.2.3 Finnish WordNets used

Multiple Finnish WordNets were used. Their codes are: `qf2`, FiWN version 2; `fin`, FiWN as included in OpenMultiWordNet (Bond and Paik 2012) which is based upon FiWN version 1, but lossily modified to promote the lemma level links explained in Section 3.2.2 to synset level links; and `qwf` which was created by Bond and Foster (2013) based on data from Wiktionary.

3.3 Obtaining sense tagged tokens

The STIFF pipeline, which starts from a bilingual corpus and ends up with sense tagged tokens is based on a generate and filter technique. First, all possible tags are generated. Next, each possible tag is annotated with different types of evidence which support it. Later stages in the pipeline then choose which tags to keep based on their supporting evidence. This means

14. Made available at `https://github.com/frankier/fiwn`.

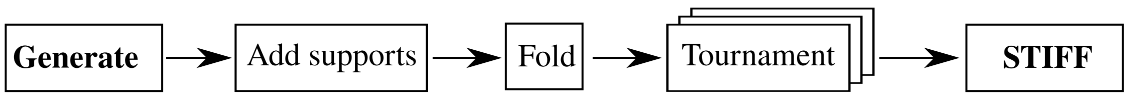


Figure 7: The overall pipeline of STIFF.

different choices for how to consider the evidence for each annotation can be generated by recombining later pipeline stages in different ways. The pipeline is shown in Figure 7.

During the generate stage, lemmas are extracted from the text and matched with potential WordNet lemmas, as described in sections 3.1 and 3.2. After this, supports are added by matching lemmas in various ways. Next, information about one language is removed, and any relevant information folded onto the other. Finally, a cascade of tournaments removes certain annotations when other annotations out rank them in some criteria.

3.3.1 Adding supports

Different types of support can be added to annotations. To add the supports for each sentence, first all relevant synsets from Finnish and Mandarin are taken $synsets_{fin}$ and $synsets_{cmn}$. The intersection $synsets_{fin} \cup synsets_{cmn}$ then corresponds to the set of annotations that receive support. When adding a synset annotation to either language, word alignments are referred to, and if an alignment aligns two words, all annotations of these aligned words with matching synsets are marked as aligned.

To improve recall, relations in WordNet can be expanded so that we can obtain some set of extra, less well supported synsets. That is we consider:

$$\text{expand}(synsets_{fin}) \cup synsets_{cmn} / synsets_{fin} \cup synsets_{cmn}$$

Currently, the only relationship type expanded in STIFF are derivational morphological links. Princeton WordNet has the most complete set of these links, so these are used. This means that English lemmas are used as an intermediary. So a synset is expanded by first fetching all of its English lemmas, then following derivational links to obtain another set of English lemmas, and then finally finding their synsets.

The reason for focussing in on derivational morphological relationships is that they tend to connect the same concept through different POSs. Due to grammar differences, as well as differences between the idiomatic ways of expressing a concept in different languages, it is not unusual that a concept may change POS during translation between languages. A noteworthy caveat here is that the ability of this scheme to expand these cross POS relations is limited by the structure of English derivational morphology.

3.3.2 Tournament stages

Now that the annotations are tagged with supports, several tournament stages can be applied. These can be combined in different ways to obtain different points on a precision/recall curve. The stages consist of:

1. Ranking tournaments, in which all annotations for each span are considered. Annotations are compared, most often pairwise, to determine whether some subset is preferable to some other subset. When this is the case the members of the former subset is said to dominate the members of the latter, and only the dominant annotations are kept.
   - Based on grammatical information of the target:
     - **pos-dom** Prefer tokens which have the correct POS according to FinnPOS.
     - **lemma-dom** Prefer tokens which have the correct lemma according to FinnPOS.
     - **recurs-dom** Prefer tokens not generated by recursive lemmatisation.
   - Based on the lexical or graphical information of the target:
     - **freq-dom** Keep only most frequent annotation.
     - **wiki-trg-dom** Prefer annotations with support from WordNets other than `qwf` over those with only support from `qwf`.
     - **hyp-dom** Where there is a synset and its hypernym ancestor, prefer the hypernym.
   - Based on the transfer process:
     - **sup-dom** Prefer tokens with some support to those with none.

- **deriv-dom** Prefer non-derived token to derived tokens.
- **align-dom** Prefer aligned tokens to non-aligned tokens.

• Based on the lexical or graphical information of the source:

- **wiki-src-dom** Prefer annotations with support from WordNets other than `qwc` over those with only support from `qwc`.
- **src-span-dom** Prefer annotations with some source A which spans some source B to those that have source B.
- **src-len-dom** Prefer annotations with some source A which has a longer length than some source B to those that have source B.

• Filtered ranking tournaments, where only a subset of annotations participate in the tournament, and others are left as they are.

- **sup-freq-dom** This is freq-dom filtered by sup-dom. It keeps the most frequent annotation for those annotations which have some support.

• **___-span:** Remove annotations outspanned by other annotations.

- **char-span-dom** Prefer annotations which outspan others on the character level.
- **tok-span-dom** Prefer annotations which outspan others on the token level.

2. Removal of annotations which are heuristically likely to be incorrect.
   - **rm-pos-norm** Remove annotations covering tokens with certain POSs according to FinnPOS. The justification is that these POSs are not included in FiWN. Removes pronouns, numerals, interjections, conjunctions, particles, punctuation and nouns which are tagged as proper nouns.
   - **rm-pos-agg** As above but additionally remove adpositions. The reason for not always removing these is some adpositions are included in FiWN, tagged as adverbs.
   - **rm-pos-soft** As above, except only remove pronouns.
   - **rm-ambg** Remove ambiguous annotations altogether.

- **-rm/-dom** Many ranking tournaments can be reinterpreted as removing any possible tournament loser, without requiring a winner exists. These are denoted **-rm** instead of **-dom**, e.g., **recurs-rm** instead of **recurs-dom**.

3.4 A second sense tagged corpus: EuroSense

EuroSense (Bovi et al. 2017) is a multilingual sense tagged corpus, obtained by running the knowledge based Babelfy (Moro, Raganato, and Navigli 2014) WSD algorithm on multilingual texts. To use this corpus in a way which is compatible with the maximum number of systems and in line with the standards of previous evaluations, it first has to be preprocessed. The preprocessing pipeline is shown in Figure 8.

Before anything else, the *fix up* stage is run. This stage is run in order to try and fix some problems found with the EuroSense 1.0 release[15]. The sub pipeline for this stage is shown in Figure 9. The *drop unanchorable* stage drops any annotations which have anchors which do not occur in any texts. The *reorder cognates* stage corrects a common error which occurs when annotations with the same anchor text end up with the wrong language tag (these are often cognates: the same word in different languages). The reordering is performed using the fact that the annotation elements seem to occur in the same order as their anchors within the ordered text elements within the eXtensible Markup Language (XML) document. Other problems are fixed in the *re tag languages* stage which works by looking for annotations whose anchors can not be found within the corresponding text. These often have the incorrect language tag and thus simply need to be associated with another text. A search to find a new anchor point and thus language is performed between two anchor points: those of the prior and subsequent correctly anchored annotations. The first possible match is always chosen. Finally, the *remove empty* stage removes any annotations which are left empty as a result of prior processing. Counts and proportions of annotation problems, both fixable and not are summarised in Table 7. As per the table, less than 1 in 10,000 annotations are dropped by this fixing process.[16]

15. Obtained from `http://lcl.uniroma1.it/eurosense/`.
16. The fixing script and fixed corpus is made available at `https://github.com/frankier/eurosense`.

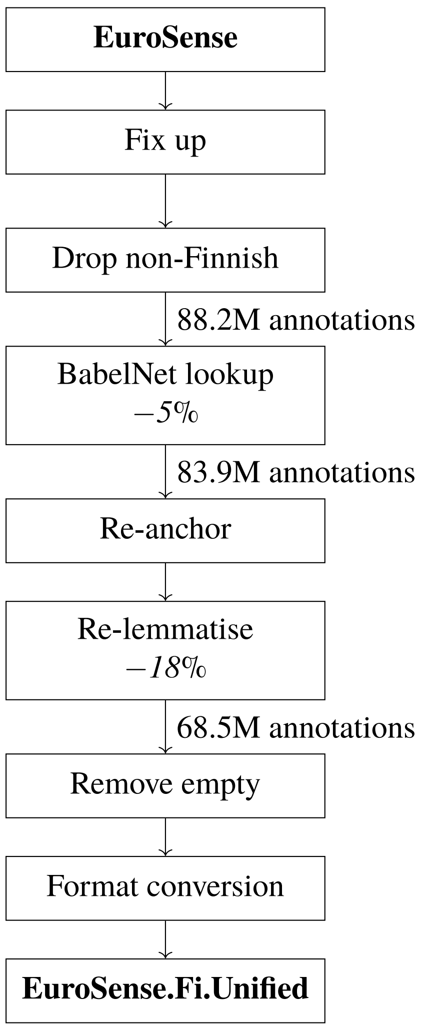


Figure 8: A diagram showing the pipeline to convert EuroSense to the unified format used for training and evaluation data.

Table 7: EuroSense problems automatically fixed. Figures are given for the high precision and high recall variants of EuroSense as counts and proportions. Columns detail: the number of annotations which could not be found within any text; the number of cognates reordered in the *reorder cognates* stage; and the further problems encountered in the *re tag languages* stage; how many of these could not ultimately be fixed.

Variant	Unanchor	Reorder	Problems	Unfix
Precision	257	3,350,417	8,247,578	3738
(%)	0.00	2.72	6.71	0.00
Coverage	426	5,643,646	30,203,097	6374
(%)	0.00	2.61	13.99	0.00

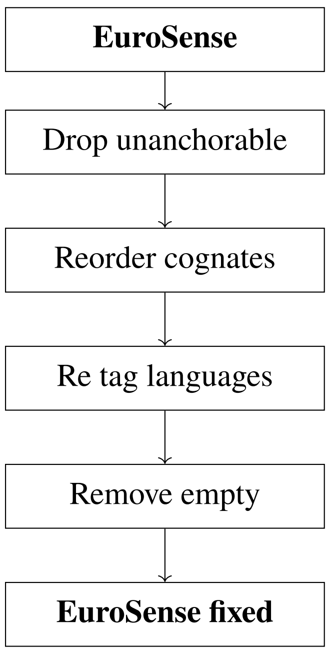


Figure 9: A diagram showing the pipeline to fix problems identified with EuroSense 1.0.

In the first stage, *drop non-Finnish*, all non-Finnish text and annotations are removed from the stream. EuroSense is tagged with synsets from the BabelNet LKB (Navigli and Ponzetto 2012). This knowledge base is based on the WordNets of many languages enriched and modified according to other sources, such as Wikipedia and Wiktionary. However, here the LKB to be used is FiWN. A mapping file was extracted from BabelNet using its Java API and a local copy, obtained through direct communication with its authors[17]. The *Babelnet lookup* stage applies this mapping. The stage will drop annotation which do not exist in FiWN according to the mapping. A BabelNet synset can also map to multiple FiWN synsets, and in this case, an ambiguous annotation can be produced.

The *re-anchor* and *re-lemmatise* stages clean up some problems with the grammatical analyses in EuroSense. EuroSense anchors sometimes include help words associated with certain verb conjugations, for example, negative forms, e.g., “ei mene”, or the perfect construction “on käynyt”. *Re-anchor* removes these words from the anchor, taking care of the cases in which the whole anchor could actually refer to a lemma form in WordNet, e.g., “olla merkitystä”. *Re-lemmatise* checks that the current lemma is associated with the annotated synsets in FiWN. In case there are no matching synsets, we look back at the surface form and check

17. Made available at `https://github.com/frankier/babelnet-lookup`.

Corpus	Genre	Original language	Sentences	Instances
OpenSubtitles2018 (STIFF)	Film and TV series subtitles	Film language — usually not Finnish	40	120
EuroParl (EuroSense)	Parliamentary proceedings	Speaker language — usually not Finnish	9	127
Total			49	247

Table 8: Scope of manual sense annotation work.

all possible lemmas obtained from Omorfi (Pirinen 2015a)[18] for matches against FiWN. At this point, any annotations which do not have exactly one lemma and one synset which exist in FiWN are dropped. In the penultimate stage, *remove empty*, any sentences without any annotations are removed entirely. Finally, the XML format is converted from the back-off annotations of the EuroSense format to the inline annotations of the unified format of Raganato, Camacho-Collados, and Navigli (2017).

3.5 Creating a manually annotated corpus

To assist with evaluating the quality of the automatically created corpora, and to provide a reliable way of assessing the quality of WSD systems, a manually annotated corpus was created. The annotation was performed jointly by the author, who is a Finnish learner at the A2 level, and a native Finnish speaker with no linguistic background. The manually annotated corpus was created by first annotating 40 sentences of the test section of STIFF, and then annotating enough sentences from test section of EuroSense that at least as many instances were annotated for EuroSense as for STIFF. The scope of the work is summarised in Table 8.

3.5.1 Annotation software

Linguistic annotation is repetitive and time consuming. Special purpose annotation software could potentially speed up the process by reducing the number of repeated actions. Existing software for word sense annotation includes: Stamp (Hovy et al. 2006); Satanic (Passonneau

18. As obtained from `https://github.com/flammie/omorfi`.

et al. 2012); and WebAnno (Yimam et al. 2013). Of these, only WebAnno is publicly available at the time of writing. While WebAnno has been used for sense annotation previously, there is no documentation available explaining how to create a word sense tag set. Thus, no usable word sense annotation software was found.

Because the amount of annotation to be done was relatively modest, and there is no requirement for advanced features like multi user annotation, a quite simple system sufficed. The annotation was created by writing a program, based on the same library code as STIFF, to generate an XML file with all possible annotations for a corpus. The human annotators then picked the correct annotations by deleting all other annotations.

So that the annotators could choose between senses, information was added to the XML file as a comment. A sample of an annotated sentence is given in Figure 10. Of note is that each annotation tag contains the lemma (wnlemma) along with the POS as well as information about the token it comes from and the WordNets the lemma comes from. Each comment contains the synonyms in Finnish and English, the definition in English, the direct hypernym, the root hypernym and the lexicographer file.

3.5.2 Annotation set up and guidelines

Annotation guidelines were created during the annotation process, as and when annotation issues arose. The final guidelines were:

- Make sure not to give word types that aren't in WordNet an annotation. This includes pronouns, modal verbs and verbs that are part of a tense e.g., participles "on ollut".
- Don't annotate with an annotation obtained only from `qwf` unless it is the only possible candidate.
- Pay attention to the English lemmas in the synset. Prefer annotations where the English is a good translation. The reason this is important is that FiWN suffers from shifts in meaning due to the fact it is created through an inherently subjective translation process, where various types of semantic shifts are possible (Osimo 2008).
- Make sure to pay attention to the lemma, POS and lexicographer file.
- If there is no good annotation available, remove all annotations.

```
<sentence id="fi/1920/10323/6416158.xml.gz
 ↪  zh_cn/1920/10323/5937327.xml.gz; 10323; 17">
  <text id="fi-tok" lang="fi">Hyvä ystäväni Alan .</text>
  <annotations>
    <annotation type="man-ann" lang="fi" anchor="Hyvä"
     ↪   anchor-positions="from_id=fi-
     ↪   tok&amp;char=0&amp;token=0" lemma="Hyvä"
     ↪   wnlemma="hyvä" wordnets="fin qf2"
     ↪   lemma-path="whole">dear.s.02 hyvä.s.18</annotation>
    <!-- dear, good, hyvä, läheinen, near, rakas, tärkeä:
     ↪   with or in a close or intimate relationship (root:
     ↪   dear.s.02; lexname: adj.all) -->
    <annotation type="man-ann" lang="fi" anchor="ystäväni"
     ↪   anchor-positions="from_id=fi-
     ↪   tok&amp;char=5&amp;token=1" lemma="ystäväni"
     ↪   wnlemma="ystävä" wordnets="fin qf2 qwf"
     ↪   lemma-path="whole">friend.n.01
     ↪   ystävä.n.03</annotation>
    <!-- frendi, friend, kaveri, ystävä: a person you know
     ↪   well and regard with affection and trust (hyp:
     ↪   person.n.01; root: entity.n.01; lexname: noun.person)
     ↪   -->
  </annotations>
</sentence>
```

Figure 10: A manually annotated snippet from OpenSubtitles2018.

Corpus	Notes	Sentences skipped	Idiom	Ambiguities	Time taken (hours:mins)
OpenSubtitles2018 (STIFF)	18	3	4	10	4:35
EuroParl (EuroSense)	8	0	5	10	5:45
Total	26	3	9	20	10:20

Table 9: Results of manual sense annotation work.

- If there is an idiom which cannot be annotated, note this and move on.
- If there is not enough context to disambiguate, note this and move on. Later this sentence will be discarded.
- Try and pick exactly one annotation for each word. If there are multiple equally suitable annotations, then:
  - If they are related by hypernymy take this into account and choose the one at the appropriate level of specificity given the context.
  - Otherwise, both can be kept, but this should only be done rarely and as a last resort.

3.5.3 Annotation results

The results of the annotation work are summarised in Table 9. The table includes counts of annotator notes, which were often used to indicate an annotation problem had occurred. These are broken down into two common cases: the case of skipping a sentence due to lack of context and the case of encountering an idiom that is not included in FiWN. Separately, ambiguous annotations are counted.

3.6 Evaluation of STIFF and EuroSense

Several variants of STIFF are made, by combining the tournament stages introduced in Section 3.3.2 in different ways. They are referred to by code names, introduced shortly. They are arranged into a tree of ablation tests, where each child variant introduces some change to its parent. The tree is given in Figure 12. The full sequence of operations for each terminal variant is given in Figure 11.

Each variant is evaluated against the manually annotated corpus of Section 3.5. The results are plotted on a precision/recall plot in Figure 13. STIFF variants on the Pareto front are highlighted as blue.

The high precision and high coverage versions of EuroSense are shown on the same plot as EP and EC respectively. There is much less direct comparability between the STIFF variants and the EuroSense variants than within their own groups. The biggest difference is that the statistics are calculated based on two different gold standard corpora. Differences between the scores can occur due to sampling, e.g., STIFF could get lucky and be evaluated based upon a part where it did well and EuroSense could get unlucky and get evaluated based upon just the part where it did poorly. This is particularly relevant here since the gold standard is relatively small. Differences between the scores could also be due to difference in the domains. The domains of film subtitles and parliamentary proceedings are quite different and it could be that one is simply easier than the other. For example, it could be that if we found Chinese-Finnish parliamentary proceedings, then the techniques behind STIFF may perform quite differently when compared to film subtitles. What is perhaps somewhat more comparable between groups is the trade off between precision and recall different variants make. Two lines with a constant ratio of precision/recall are shown through EP and EC.

We start by comparing the results of the evaluation of the Finnish section of EuroSense performed here with the evaluation of its original authors. Bovi et al. (2017) evaluated 50 sentences each from the English, French, German and Spanish sections of EuroSense and found precisions of 81.5%, 71.8%, 89.3% and 82.5% respectively for the high precision variant. The corresponding result obtained here (EP) is just 43.6%, however, there are significant differences in evaluation method which make direct comparison difficult. Firstly, Bovi et al. (2017) used two independent annotators to produce two annotated corpora with a disagreement rate of around 15% as compared to the annotation team approach used here which results in a single annotated corpus. It is not reported how Bovi et al. (2017) handled disagreements. One possibility is that they are discarded, which would prevent more difficult cases from being evaluated, boosting precision. More likely, however, is that when a disagreement occurs, a

TP was be awarded to EuroSense when it matched either of the senses chosen the two annotators, which would also boost the resulting precision. Secondly, the change of sense inventory from BabelNet to WordNet performed in Section 3.4 is likely to make quite a big difference. BabelNet contains far more named entities — proper nouns that are likely to have an entry in an encyclopedia rather than those words likely to have an entry in a dictionary — than WordNet. These items have fewer senses, often just one, and therefore changing sense inventory from BabelNet to WordNet drops these easy cases from consideration, lowering precision. Thirdly, the need for the fixing process described on page 54 brings up further questions such as whether the version of EuroSense the authors performed their evaluation against had the same problems and if not, whether their evaluation can be said to be valid for the publicly available version of EuroSense.

We now compare variants of STIFF starting from the unambiguous variant (U), which removes all ambiguous annotations, leaving only those which were not ambiguous to begin with. We then go down the first branch of Figure 12, attempting to improve the precision with the monolingual precision (MP_) systems. These improve precision by concentrating on improving the quality of the target language lemmas; They take no account of data from the source language. The bilingual precision (BP_) variants start to trade off precision for recall by adding in tournaments based on supports from the source language.

The next branch starts with a high monolingual recall system (MR), which just selects the most frequent annotation for a span. The bilingual recall (BR_) systems take account of support from the source language and progressively attempt to trade off decreasing recall for increasing precision. Together the precision and recall series show it is possible to trace out a Pareto front by progressively making adjustments to the stages which make up the different variants.

Ultimately, the BP4 system is selected for usage in Chapter 4. Firstly, it is on the Pareto front, a necessary criterion for selection. Secondly, keeping in mind the caveats of comparing between the two groups of systems, it seems like it may have a similar level of precision and recall as EP, or at very least represent a similar choice in terms of trading off between precision and recall to it. This is useful since this is the variant of EuroSense used in Chapter 4, and thus choosing BP4 allows for a more direct comparison.

U:

$$\begin{tikzcd}\text{rm-ambg}\end{tikzcd}$$

BP3:

$$\begin{tikzcd}\text{recurs-rm} \arrow[d] \\ \text{rm-pos-agg} \arrow[d] \\ \text{lemma-rm} \arrow[d] \\ \text{sup-dom} \arrow[d] \\ \text{align-dom} \arrow[d] \\ \text{deriv-dom} \arrow[d] \\ \text{rm-ambg}\end{tikzcd}$$

SP:

$$\begin{tikzcd}\text{sup-rm} \arrow[d] \\ \text{rm-ambg}\end{tikzcd}$$

BP5:

$$\begin{tikzcd}\text{recurs-rm} \arrow[d] \\ \text{rm-pos-agg} \arrow[d] \\ \text{lemma-rm} \arrow[d] \\ \text{sup-dom} \arrow[d] \\ \text{align-dom} \arrow[d] \\ \text{deriv-dom} \arrow[d] \\ \text{src-span-dom} \arrow[d] \\ \text{sup-freq-dom} \arrow[d] \\ \text{wiki-trg-dom} \arrow[d] \\ \text{hyp-dom} \arrow[d] \\ \text{rm-ambg}\end{tikzcd}$$

BR4:

$$\begin{tikzcd}\text{sup-dom} \arrow[d] \\ \text{align-dom} \arrow[d] \\ \text{rm-pos-agg} \arrow[d] \\ \text{pos-dom} \arrow[d] \\ \text{lemma-dom} \arrow[d] \\ \text{freq-dom} \arrow[d] \\ \text{rm-ambg}\end{tikzcd}$$

SR:

$$\begin{tikzcd}\text{sup-rm} \arrow[d] \\ \text{freq-dom} \arrow[d] \\ \text{rm-ambg}\end{tikzcd}$$

MXP:

$$\begin{tikzcd}\text{sup-rm} \arrow[d] \\ \text{align-rm} \arrow[d] \\ \text{rm-pos-agg} \arrow[d] \\ \text{lemma-rm} \arrow[d] \\ \text{rm-ambg}\end{tikzcd}$$

Figure 11: Pipelines showing how STIFF stages are combined to form the different variants. Only the terminal variants are shown.

```
U
├─ MP1    +recurs-rm, +rm-pos-norm, +lemma-dom
│  └─ MP2    rm-pos-agg / rm-pos-norm
│     └─ MP3    lemma-rm / lemma-dom
│        └─ BP1    +sup-dom
│           └─ BP2    +align-dom
│              ├─ BP3    +deriv-dom
│              └─ BP3A    +sup-freq-dom
│                 └─ BP4    +wiki-trg-dom
│                    └─ BP5    +deriv-dom, +src-span-dom, +hyp-dom
├─ MR    +freq-dom
│  └─ BR1    +sup-dom, +align-dom, +pos-dom, +lemma-dom
│     └─ BR2    +rm-pos-soft
│        └─ BR3    rm-pos-norm / rm-pos-soft
│           └─ BR4    rm-pos-agg / rm-pos-norm
├─ SP    +sup-rm
├─ SR    +sup-rm, +freq-dom
└─ MXP    +sup-rm, +align-rm, +rm-pos-agg, +lemma-rm
```

Figure 12: A tree providing an ablation lineage structure to the STIFF variants presented here. + denotes the addition of a stage while x / y indicates replacing y with x. Since operations are not reordered, the order can be found from the terminal node by referring to Figure 11.

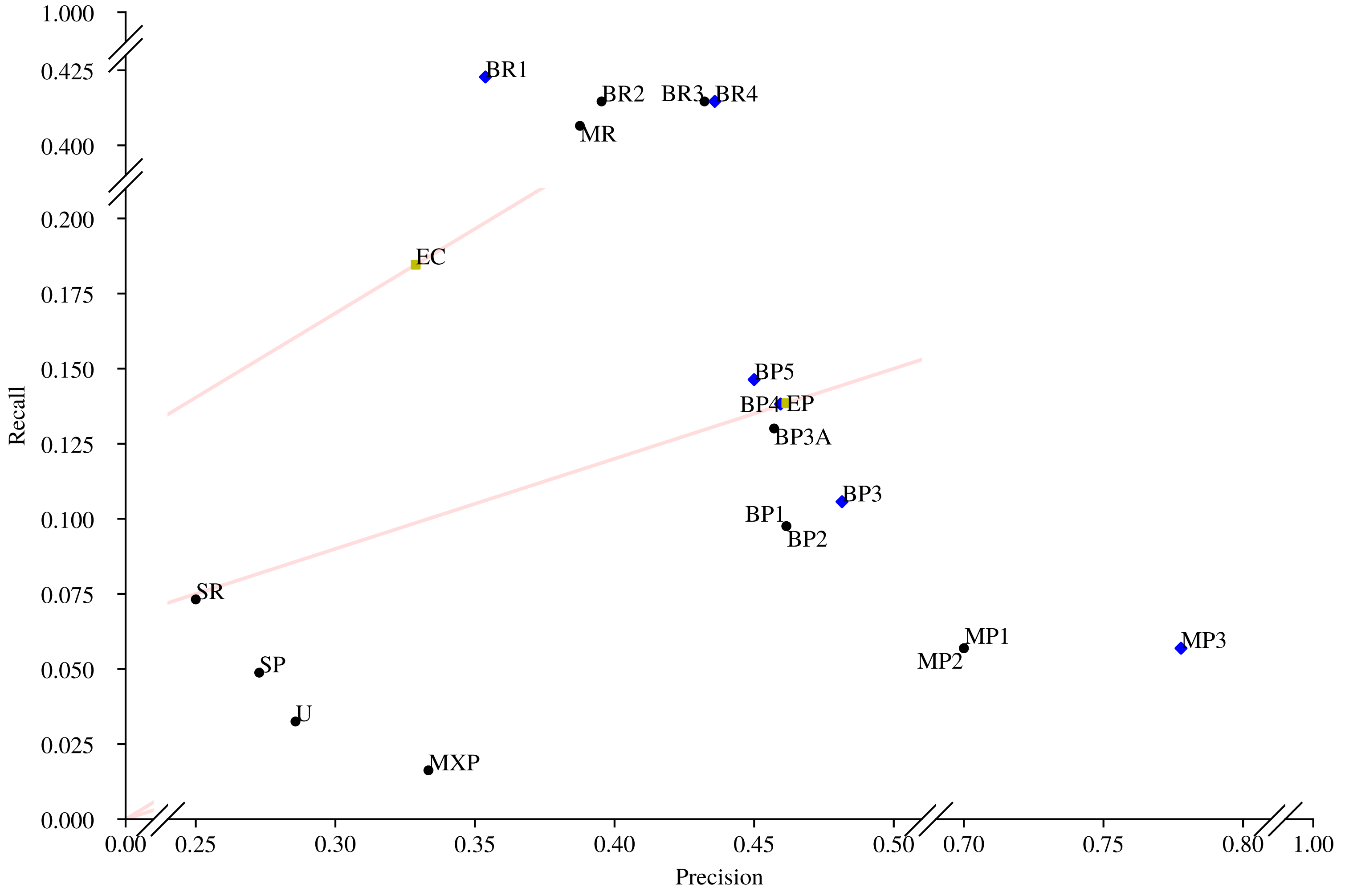


Figure 13: A precision/recall plot showing variants of STIFF alongside the processed version of EuroSense. Note the discontinuities in the axes. The pink lines indicate lines of constant precision/recall ratio, passing through EC and EP. STIFF variants on the Pareto front are highlighted in blue, while EuroSense variants are coloured yellow.

4 Implementation of WSD techniques

This chapter is based on, and presents an extended version of, the evaluation of Robertson (2019). The evaluation has been extended to take full advantage of the resources developed in Chapter 3, including the STIFF automatically induced corpus and the manually annotated corpus. Additional systems based on contextual word vectors derived from language modelling have been added. Finally, experiments based on training on English language data have been performed.

This evaluation is based on the all words variant of the Word Sense Disambiguation (WSD) task. In this task, the aim is to identify and disambiguate all words in some corpus. This is contrasted with the lexical sample approach, where a fixed set of words are chosen for evaluation. As we have seen in Section 2.3, there are many systems and approaches which have been proposed for performing WSD. To select techniques for this evaluation, the following criteria were used:

- Prefer techniques which have been used in previous evaluations for English.
- Prefer techniques with existing open source code that can be adapted, or with existing resources which can be downloaded to avoid a lengthy training step.
- Apart from this, include also simple schemes, especially if they represent an approach to WSD not covered otherwise.

The last two criteria have led to the inclusion of multiple techniques based upon representation learning, where some representation of words or groups of words is learned in an unsupervised manner from a large corpus. To perform WSD based on these representations, a relatively simple classifier, such as a nearest neighbour classifier is then used. This approach to WSD additionally acts as a grounded extrinsic evaluation of the quality of the representations.

Table 10: Basic information about the corpora and their sections as used in this evaluation.

Corpus	Section	Type	Sentences	Instances	Synsets
EuroSense	train	Automatic	1,564,967	6,675,180	30,814
	dev	Automatic	991	4461	1463
	test	Manual	9	117	102
STIFF BP4	train	Automatic	9,179,742	8,216,011	34,481
	dev	Automatic	960	525	332
	test	Manual	40	95	85
SemCor	train	Manual	37,176	226,695	25,916
WordNet (v3.1)		LKB			117,659
FinnWordNet (v2)		LKB			120,449

4.1 Resources

In order to conduct a WSD evaluation, we need a Lexical Knowledge Base (LKB), which provides the word senses themselves and an evaluation corpus, for scoring the different systems against. For supervised systems, we additionally need a training corpus. These are covered in Section 4.1.1.

The current generation of Natural Language Processing (NLP) systems make copious usage of word embeddings, and other resources which compress massive raw corpora into a form which is directly useful form as a lexical resource such as language models, as do some of the systems evaluated here. Section 4.1.2 goes into more depth about the particular word embeddings and language models used in this evaluation.

4.1.1 LKBs and corpora

The corpora used in this evaluation are already described to some extent in Chapter 3. In particular, the STIFF automatically induced sense annotated corpus is developed in Section 3.3, the manually annotated corpus is developed in Section 3.5, and the Finnish section of the EuroSense corpus (Bovi et al. 2017) is described in Section 3.4. In terms of LKBs, the FinnWordNet (FiWN) LKB (Lindén and Carlson 2010) first mentioned in Section 2.2.1 and extended with pseudo word frequency data in Section 3.2.2 is used. In some experiments, the manually sense annotated English language corpus SemCor (George A. Miller et al. 1994) is used as training data. Within this section, an English language evaluation corpus, SemEval-

ALL, compiled from the evaluation corpora of SENSEVAL and SemEval WSD shared tasks by Raganato, Camacho-Collados, and Navigli (2017) is included as an extra point of comparison.

The process by which the corpora are divided into segments is described in more detail in Section 4.2.1. Basic statistics about all the corpora and their segments are given in Table 10: the number of sentences, how many sense tagged instances there are in the corpus segment and how many unique synsets there are. The number of synsets in WordNet and FinnWordNet are given for comparison: in all cases, only a fraction of synsets available in the LKBs are tagged.

We now turn to the matter of the distribution of senses within the different corpora and LKBs since this can make a significant difference to the difficulty of an evaluation corpus and the usefulness of a training corpus as well as the overall difficulty of performing WSD on an LKB. In particular, we look at different measures of ambiguity and information entropy. Ambiguity is the number of senses within a confusion class, defined here as arising from attempting to disambiguate a lemma, Part Of Speech (POS) pair. Information entropy, measured in bits, is defined as:

$$H(\gamma) = -\sum_{s \in \gamma} \gamma(s) \log_2(\gamma(s))$$

Given a discrete probability distribution γ, information entropy can be used as a measure of skew in a distribution. A uniform distribution will have an entropy of $\log_2(|\gamma|)$, whereas a distribution with all its probability mass on a single s will have an entropy of 0.

Now we consider different ways of aggregating these measures across whole corpora and LKBs. The values of the resulting aggregates are shown in Table 11. We can average the entropy per lemma, POS pair, abbreviated as H or we can weight by the number of occurrences in the corpus, denoted WH. We can then consider ambiguity A, which is how many senses are available in the corpus for each lemma, POS pair and its corresponding weighted version WA. Finally, there is lexical ambiguity LA, which is how many senses are available from the LKB for each lemma, POS pair and its weighted version WLA. The lexical ambiguity can be given directly for the LKBs. The entropy figures for the LKBs are based on the count data included with Princeton WordNet (PWN) and transferred to FiWN in Section 3.2.2.

Table 11: Sense distribution information about training and test corpora as used in this evaluation. *SemEval-ALL* is only used for comparison within this section and not used elsewhere in this chapter. H = entropy; WH = weighted entropy; A = (corpus) ambiguity; WA = weighted (corpus) ambiguity; LA = lexical ambiguity; WLA = weighted lexical ambiguity

Corpus	Language	H	WH	A	WA	LA	WLA
EuroSense train	Finnish	0.39	1.04	2.1	8.0	2.2	8.6
STIFF BP4 train	Finnish	0.13	0.61	1.3	5.1	3.1	9.5
SemCor	English	0.32	1.20	1.6	5.8	2.4	6.8
EuroSense test	Finnish	0.04	0.08	1.0	1.1	9.0	10.7
STIFF BP4 test	Finnish	0.01	0.02	1.0	1.0	7.2	7.5
SemEval-ALL	English	0.26	0.59	1.4	2.1	4.3	5.8
FinnWordNet	Finnish	0.31	1.12			1.5	
WordNet	English	0.28	1.02			1.3	

Pasini and Camacho-Collados (2018), give values of two similar measures for many English language annotated corpora. However, their figures do not entirely agree with the ones given here. Their figures for ambiguity are equivalent to and agree with those LWA figures given here. However, their entropy figures, which should be equivalent to H here, given for SemCor and SemEval-ALL are 0.27 and 0.18 respectively. The reason for this discrepancy is unclear.

We now analyse the figures from Table 11 in more detail. We are particularly in figures which tell us about the coverage and therefore the quality of our training corpora and measures which tell us the difficulty of our testing corpora. We begin by noting that the training segment of STIFF has a low H and WH compared to the training segment of EuroSense and SemCor, meaning it has a skewed coverage of the senses of each lemma. Concordantly, the gap between WA and WLA is relatively small for the training segment of EuroSense and SemCor — 0.6 and 1.0 respectively — compared to the training segment of STIFF — 4.4. This suggests that STIFF has relatively poor coverage of the different senses of the lemma, POS pairs it includes. We now move from considering only the mean of the entropy distribution of a corpus to considering the whole distribution. Figure 14 shows the distributions of the entropy of the sense distributions of different instances of the training corpora — the mean of these distributions is WH. Noting the different y-axes, it becomes apparent that a large part of the reason for the low mean WH of STIFF is that it has about twice the number of instances with 0 entropy. These instances are necessarily part of a confusion class where there is exactly one

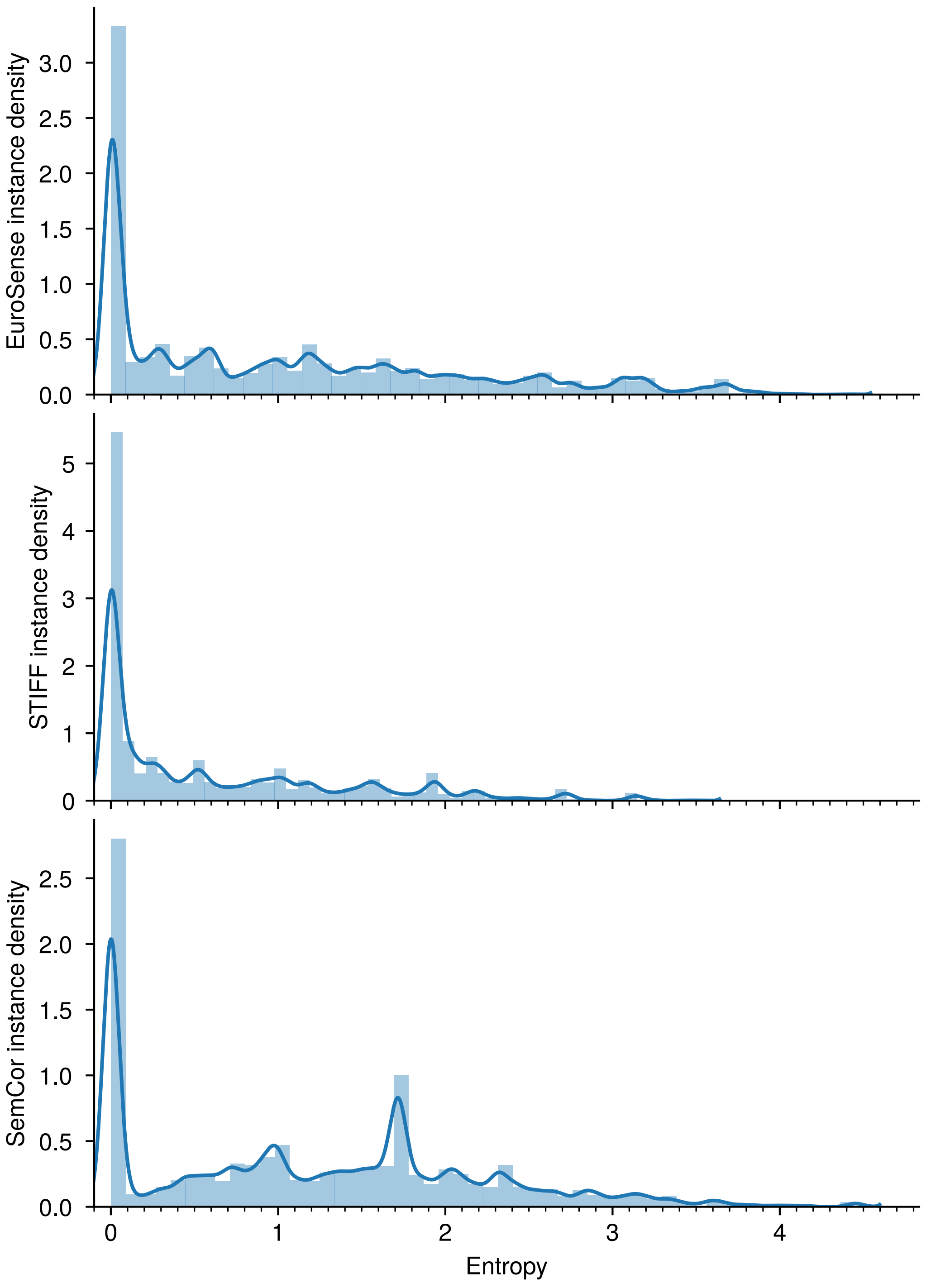


Figure 14: Distribution plots with histogram and kernel density estimate showing the entropy distributions of instances in within training corpora. Note the separate y-axes.

sense available in the training corpus. This means a supervised system based upon the word expert assumption[1] has no ability to associate different contexts with different senses, and is instead forced to always tag with this sense.

Figure 15 shows a histogram of lexical ambiguity levels of lemmas, POS pairs in FiWN and PWN, corresponding to LA in Table 11. Figure 16 shows two histograms of the weighted lexical ambiguity in the manually annotated test segments of the Finnish language STIFF and EuroSense and the English language subcorpora of SemEval-ALL, corresponding to WLA in Table 11. Both of these figures are in agreement Table 11 in showing that there is a greater degree of lexical ambiguity both in FiWN when compared to PWN and in the Finnish manually sense annotated evaluation corpora versus SemEval-ALL and its subcorpora, making them inherently more difficult for WSD systems.

4.1.2 Word embeddings and language models

Table 12 summarises the word embeddings and language models used. Due to the large number of word forms a Finnish lemma can take, it is of note here whether the word embedding represents word forms or lemmas, and if it represents word forms, whether it uses any character level information, which should help to combat data sparsity. Despite the use of subword information, none of these embeddings can analyse out of vocabulary word forms. Cross lingual word embeddings embed words from multiple languages in the same space, a property utilised in Section 4.3.3.

To extend word representations to sequences of words such as sentences, taking the Arithmetic mean of Word Embeddings AWE has been commonly used as a baseline (Arora, Liang, and Ma 2017). Various incremental modifications have been suggested. Rücklé et al. (2018) suggest concatenating the vectors formed by multiple power means, including the arithmetic mean. Variants CATP3 and CATP4 are used here. The former is the concatenation of the minimum, arithmetic mean, and the maximum, while the latter contains also the 3rd power mean. Arora, Liang, and Ma (2017) proposed Smooth Inverse Frequency (SIF), by taking a weighted average according to $\frac{a}{a+\mathrm{p}(w)}$, where a is a parameter and $\mathrm{p}(w)$ is the probability

1. See Section 2.3.4.

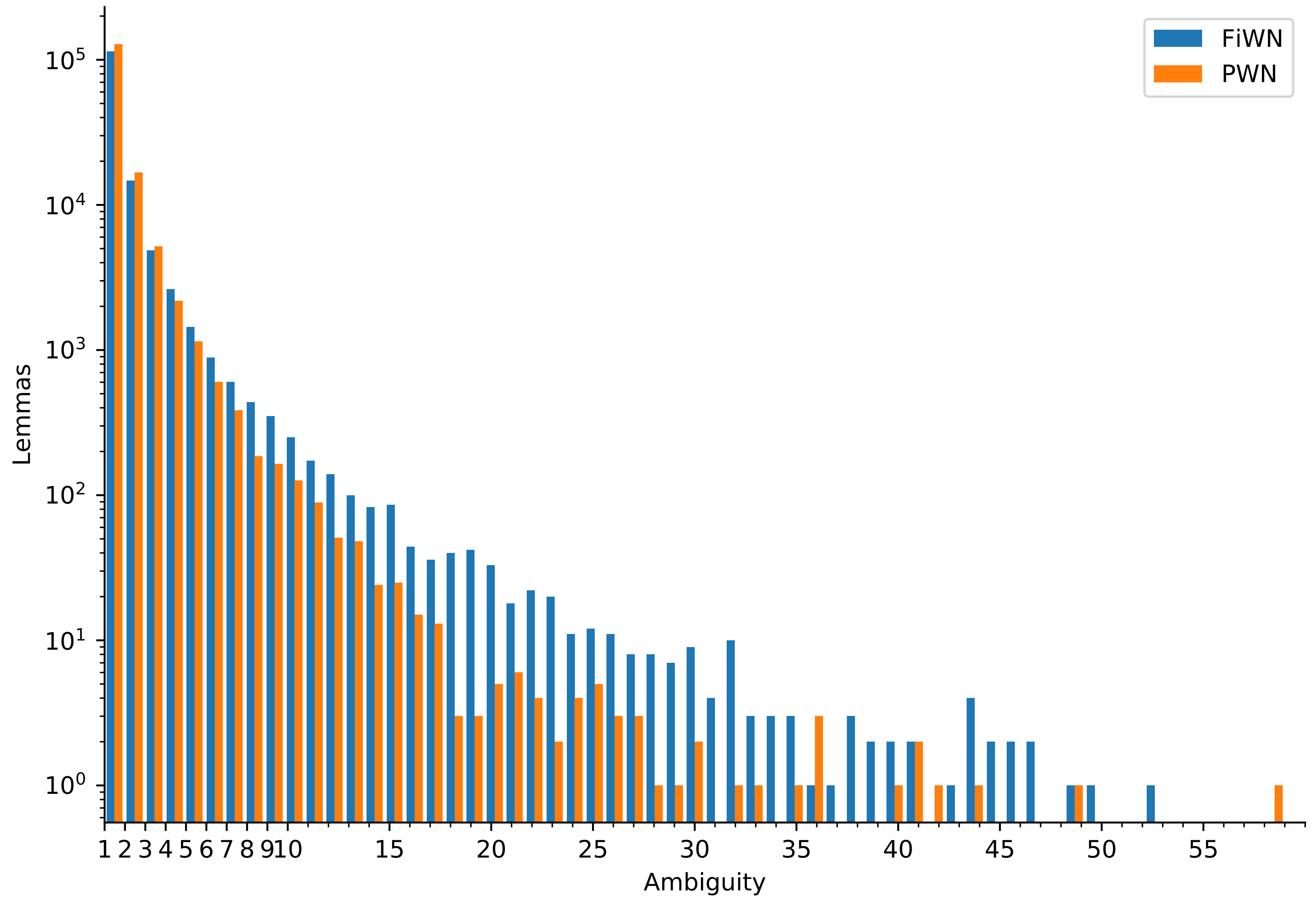


Figure 15: Histogram of ambiguity levels of lemma, POS pairs in FiWN and PWN. Note the logarithmic scale on the y-axis.

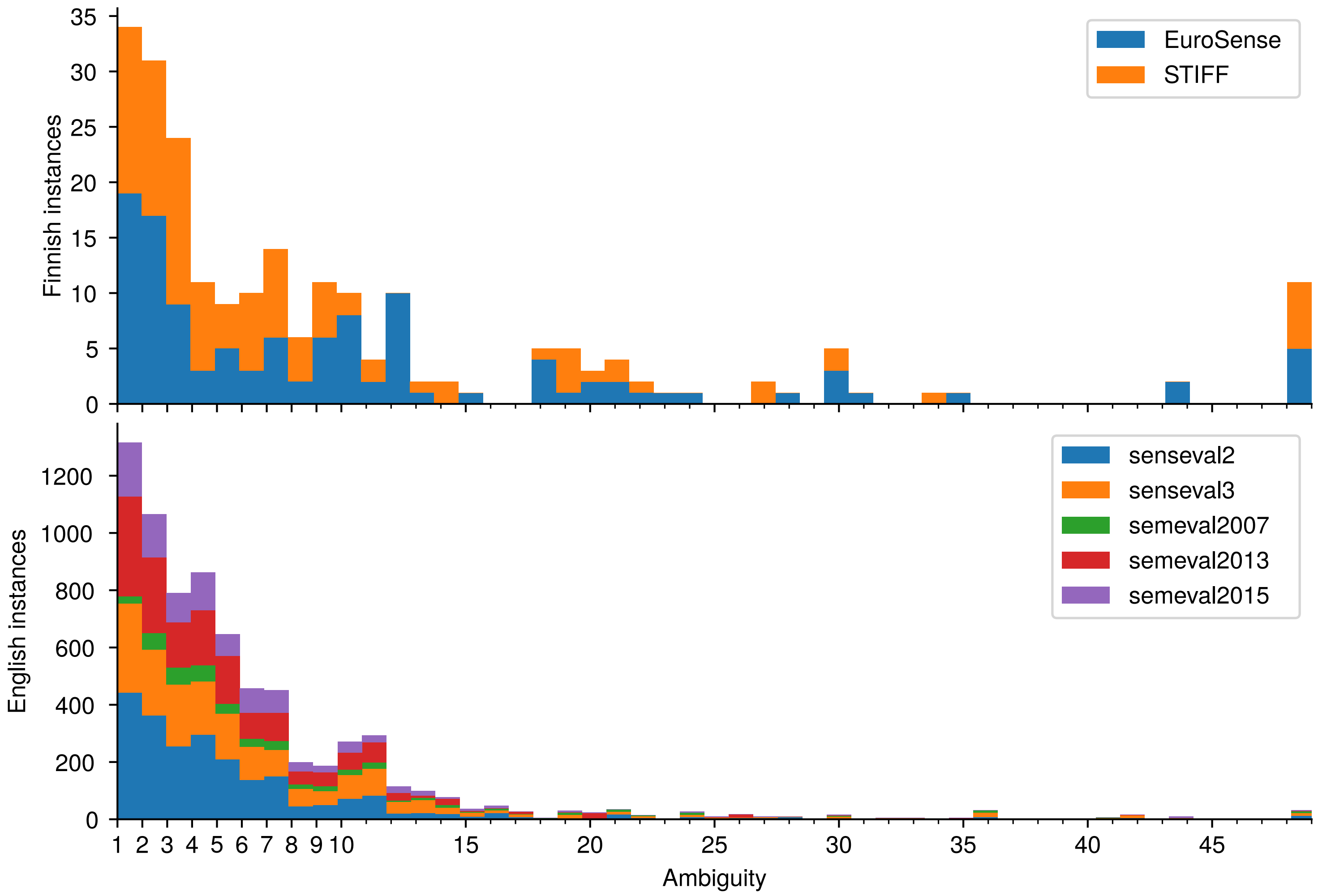


Figure 16: Histogram of ambiguity levels of lemma, POS pairs in Finnish and English testing corpora. SemEval-ALL has been broken down by the subcorpus from which it came. Note the separate y-axes.

Table 12: Word embeddings and language models used. Columns include the number of dimensions, whether the representation takes account of sub word structure and whether it is cross lingual.

Name	Type	Training data	Dim	Represents	Sub	X-ling
MUSE Supervised fastText[a,b]	Word vectors	Wikipedia & bilingual dictionary	300	Word forms	Yes	Yes
ConceptNet Numberbatch 17.06[c,d]	Word vectors	Wikipedia & ConceptNet	300	Lemmas & MWEs	—	Yes
NLPL word2vec[e,f]	Word vectors	Wikipedia & CommonCrawl[g]	100	Word forms	No	No
BERT-Base, Multilingual Cased[h]	Language model	Wikipedia	768	Word forms	Yes	Yes
Context2Vec[j,k]	Language model	CommonCrawl [g]	600	Word forms	No	No

[a] Conneau et al. (2018)

[b] `https://github.com/facebookresearch/MUSE`

[c] Speer, Chin, and Havasi (2017)

[d] `https://github.com/commonsense/conceptnet-numberbatch`

[e] Fares et al. (2017)

[f] `http://vectors.nlpl.eu/repository/`

[g] `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989`

[h] Devlin et al. (2019)

[i] `https://github.com/google-research/bert/blob/master/multilingual.md`

[j] Melamud, Goldberger, and Dagan (2016)

[k] Trained as part of this thesis. See Section 4.3.7.

of the word, and then performing common component removal. In the variant used here, (referred to as pre-SIF) a is set to the suggested value of 1×10^{-3} and common component removal is not performed, while $\mathrm{p}(w)$ is estimated based upon the word frequency data of Speer et al. (2018)[2].

4.2 Method

This section describes some of the methodological considerations taken in this evaluation.

4.2.1 Corpus division and model selection

In some cases many variants of a single system are possible. We would like to select the variant which performs best to represent each system. However, this model selection step can be considered as another training step. Therefore if we were to use our testing data to perform it, we would be essentially training on our testing data. This would mean out final scores would not be admissible since although the individual system would not have access to the instances from the testing data, it may be that we end up choosing a system particularly suited for the distribution of the testing data. Another tempting possibility is to reuse the training data, however, this would tell us nothing about how well our model generalises, only how well it can memorise the training data and is therefore likely to result in overfitting. The solution is to make a third dataset which is neither the training nor testing data set, known as the development data set.

Thus each corpus is ultimately split into three sections. First, the first 1000 sentences are split out from each corpus with the rest being training data (train). From these 1000 sentences, the manually annotated part that was annotated in Section 3.5 is deleted to form the development section (dev) which is used for model selection. Finally, the testing section is formed from the manually annotated data (test). Refer back to Table 10 for the basic information about these sections.

2. Obtained from `https://github.com/LuminosoInsight/wordfreq`.

4.2.2 Corpus preprocessing

The resulting corpus is already sentence and word segmented. Additionally, the instance to be disambiguated is passed to each system with the correct lemma and POS tag, meaning the evaluation only tests the disambiguation stage of a full WSD pipeline and not the candidate extraction or POS tagging stage. The corpus is further processed with FinnPOS (Silfverberg et al. 2016)[3] for systems that need POS tags and/or lemmas for the words in the context.

4.2.3 Significance testing

```
Function Mk-Schedule
  (random seed $s$, number of bootstrap iterations $b$, length of gold standard $n$) returns
  schedule $t$
    seed random number generator with $s$
    let $t$ be an array of dimension $b \times n$
    for $i$ from 1 to $b$ do
        for $j$ from 1 to $n$ do
            $t_{i,j}$ := a number uniformly sampled from 1 to $n$
        end
    end
end
```

Algorithm 1: Function to make a resampling schedule for the bootstrapping procedure.

```
Function Bootstrap-Gen
  (schedule $t$ of dimension $b \times n$, system results $x$ of dimension $n$, gold standard $g$)
  returns $F_1$-measure $f$, resampled $F_1$-measures $f^*$
    $f$ = $F_1$-measure of $x$ against $g$
    let $f^*$ be an array of dimension $b$
    for $i$ from 1 to $b$ do
        let $x^*$ be an array of dimension $n$
        for $j$ in 1 to $n$ do
            $x^*_j$ := $x_{t_{i,j}}$
        end
        $f^*_i$ = $F_1$-measure of $x^*$ against $g$
    end
end
```

Algorithm 2: Function to create a bootstrapped distribution of F_1-measures.

3. Obtained from `https://github.com/mpsilfve/FinnPos`.

```
Function BOOTSTRAP-CMP
 (system a's $F_1$-measures $f^a$ and $f^{a*}$,
 system b's $F_1$-measures $f^b$ and $f^{b*}$) returns directionality $c$, p-value $p$
    $d := f^b - f^a$
    if $d < 0$ then
        $d := -d$
        $c$ := system a has a larger $F_1$-measure
    else
        $c$ := system b has a larger $F_1$-measure
    end
    $s := 0$
    for $i$ from 1 to $b$ do
        if $c$ says system b has larger $F_1$-measure then
            $d^* := f_i^{b*} - f_i^{a*}$
        else
            $d^* := f_i^{a*} - f_i^{b*}$
        end
        if $d^* > 2d$ then
            increment $s$
        end
    end
    $p := \frac{s}{b}$
end
```
Algorithm 3: Function to compare two bootstrapped distributions to determine direction and significance level.

While we can calculate the F_1 score for an individual run, it may be that this somehow due to the "luck of the draw" in which a certain lucky system gains an advantage due to the test set containing more instances which are easier for it to classify correctly. Bootstrapping resamples the results from each system with replacement many times to synthesise a distribution from the sample, essentially creating an estimate of many "luck of the draw" scenarios. The probability that one system really is better than another can then be estimated by considering the distribution formed by taking the difference between the two systems' synthetic distributions and seeing how its probability mass compares to 0. The probability that we should reject the null hypothesis, that the distributions are the same, is called a p-value. The procedure used here roughly follows that of Berg-Kirkpatrick, Burkett, and Klein (2012)[4]. The bootstrapping significance testing is based on paired resampling. This means that all the results of all systems are joined or paired together and so each sample draws information about each instance from all systems. A worked example is given in Figure 17.

The specific procedure used here is outlined in detail in Algorithms 1, 2 & 3. In Algorithm 1, a list of resamplings acting as a to do list, agenda or *schedule* is built, to be used by the later steps. This schedule is reused per gold standard, meaning that in this case there are separate schedules created for STIFF and EuroSense and for the automatically annotated development sections and manually labelled test sections thereof (introduced in Section 4.2.1).

In Algorithm 2 a distribution of F_1-measures is generated by using the schedule to create many guesses and scoring them. Worth noting here is that contrary to a usual set up of calculating F_1-measure, a single instance occurring multiple in a guess will be counted multiply. A conventional scoring program, such as the one released alongside Raganato, Camacho-Collados, and Navigli (2017) and used in previous shared tasks only count True Positives (TPs), False Positives (FPs) and False Negatives (FNs) once. Here, in keeping with the idea of resampling with replacement, these are all counted multiply.

Finally, Algorithm 3 calculates the final p-value calculation. The comparison is two sided. This means the null hypothesis is that the two systems have identical F_1-measures, and rejecting it can lead to the conclusion that either system is better.

4. Note in particular the algorithm listing on page 2 and the footnote on page 3.

Iteration i of schedule, t_i

Line no.
53
74
53
...

Gold

N	*ID*	*Synset*
1	W-1	02603699-v
…	…	…
53	W-80	05611302-n
…	…	…
74	W-90	02499036-a
…	…	…

System A

N	*ID*	*Synset*
1	W-1	00941990-v
…	…	…
53	W-80	05611302-n
…	…	…
74	W-90	01038332-a
…	…	…

System B

N	*ID*	*Synset*
1	W-1	02603699-v
…	…	…
53	W-80	05611302-n
…	…	…
74	W-90	02499036-a
…	…	…

…

System Z

N	*ID*	*Synset*
1	W-1	02603699-v
…	…	…
53	W-80	06341340-n
…	…	…
74	W-90	02499036-a
…	…	…

Resampled

N	*ID*	*Gold*	*A*	*B*	…	*Z*
1	W-80	05611302-n	05611302-n	05611302-n	…	06341340-n
2	W-90	02499036-a	01038332-a	02499036-a	…	02499036-a
3	W-80	05611302-n	05611302-n	05611302-n	…	06341340-n

Figure 17: An example to illustrate how the process of paired resampling of the results from different WSD systems works for a single bootstrap iteration.

The p-value used here is 0.05. In statistical test theory, FPs and FNs are referred to as type I and type II errors respectively, while those items being compared (in this case WSD systems) are referred to as treatments. The p-value is the chance of making a type I error. When we consider that we are making multiple comparisons in our analysis, the chance that there is a type I error on one of the comparisons becomes quite probable since it is the chance of a type I error *per-comparison*. There exist various compensating procedures that attempt to bound the probability of one or more type I errors occurring across all comparisons. In line with previous NLP comparisons, no such compensation is made here.

Piepho (2004) proposes presenting the result of an all pairs comparison by ranking results and labelling each treatment with one or more letters denoting its membership of a class of treatments which are not significantly different. The implementation developed as part of this thesis finds the letters by forming an undirected graph of treatments with edges between treatments that have non-significantly different p-values and finding cliques in the graph, using the *networkx*[5] Python library.

4.3 Systems

This section describes the systems and configurations under evaluation. A simple baseline is presented first, then some knowledge based WSD systems followed by some supervised systems.

Knowledge based WSD systems use only information in the LKB. In almost all dictionary style resources, this can include the text of the definitions themselves. In WordNet style resources, this can include also the graphical structure of the LKB. Sometimes knowledge based systems incorporate frequency information. Here, where there is a possibility to do so, variants with and without are selected.

Supervised WSD systems are based on supervised Machine Learning (ML). Most typically in WSD a separate classifier is learned for each individual lemma. Another approach is to classify at the synset level. Here, all supervised systems were able to fall back to the 1st sense baseline for unseen tokens.

5. Obtained from `https://networkx.github.io/`.

4.3.1 Baseline

We can define limits to calibrate the performance of the WSD systems which also tell us about the sense distributions in our corpora and LKB. We may consider the proportion of unambiguous test instances, *unambg*. In a supervised setting, we can consider the proportion of test instances which have their lemma associated with the true sense in the training data, *known* — i.e., the maximum accuracy of a system based upon the word expert assumption.

Baselines both tell us about the sense distributions of the corpora and act as trivial WSD system. The most common WSD baseline simply picks the most frequent sense for each instance, which in many LKBs is the first numbered sense. Here the data from Section 3.2.2, is used for this *1st* baseline. Picking a random sense is another baseline, referred to as *rand*.

4.3.2 UKB

UKB (Agirre, Lacalle, and Soroa 2014) is a knowledge based system, representing the graph based approach to WSD. Since it works on the level of synsets, the main algorithm is essentially language independent, with the candidate extraction step being the main language dependent component. UKB can also make use of language specific word sense frequencies.

As noted in Agirre, López de Lacalle, and Soroa (2018), depending on the particular configuration, it is easy to get a wide range of results using UKB. The configurations used here are based on the recommended configuration given by Agirre, López de Lacalle, and Soroa (2018). For all configurations, the *ppr w2w* algorithm is used, which runs personalised page rank for each target word. One notable change here is that contexts are fixed to a single sentence, since this the same input as is given to the other systems. Variations with and without access to word sense frequency information are given, (freq & no freq) with the latter assumed to similar to the configuration given in Raganato, Camacho-Collados, and Navigli (2017).

4.3.3 Lesk with cross lingual word embeddings

A variant of Lesk — Lesk with cross lingual word embeddings — is included to represent the gloss based approach to WSD. The system is also referred to here as cross lingual Lesk and xLesk. The variant presented here is loosely based upon Basile, Caputo, and Semeraro (2014). The technique is a derivative of simplified Lesk (Kilgarriff and Rosenzweig 2000a), in that words are disambiguated by comparing contexts and glosses. For each candidate definition, the word vectors of each word in the definition text are aggregated to obtain a definition vector. The word vectors of the words in the context of the word being disambiguated are also aggregated to obtain a context vector. Definitions are then ranked from best to worst in descending order of cosine similarity between their definition vector and the context vector. Frequency data (freq) can be incorporated by multiplying the obtained cosine similarities by the smoothed probabilities[6] of the synset given the lemma.

Since the words in the context are Finnish, but the words in the definitions are English, cross lingual word vectors are required. The embeddings used are fastText, Numberbatch and the concatenation of both. Other variations are made by the choice of aggregation function, choosing whether or not to only include words which occur in a WordNet, and whether glosses are expanded by adding also the glosses of related synsets. The gloss expansion procedure applied follows Banerjee and Pedersen (2002, Chapter 6).

4.3.4 Lesk++

Oele and Noord (2017) introduced Lesk++. The system is based on word sense embeddings which are enriched with word sense vectors based on the graphical structure of WordNet using the AutoExtend retrofitting method Rothe and Schütze (2017)[7]. To create these vectors, the ConceptNet Numberbatch vectors introduced in Table 12 are used as a starting point. The final embedding contains surface lemmas, synsets and WordNet lemmas: (synset, lemma) pairs embedded in a single space.

Lesk++ works sentence wise, representing the sentence context as an average of vectors of disambiguated and undisambiguated words: surface lemma vectors for those that have not yet been disambiguated, and (synset, lemma) vectors for those that have not. Glosses are also represented using an average of word vectors. The sentence is processed from the least

6. This smoothing procedure is given in Section 3.2.2.

7. Using their software obtained `https://github.com/casaro/AutoExtend`. The preprocessing script and resulting vectors are made available at `https://github.com/frankier/AutoExtend`.

ambiguous word to the most. For each word, the sense which maximises the cosine similarity of the context vector and the sense vector and the context vector and the sense's gloss vector is chosen. Here, similar to cross lingual Lesk, there are variants which use different aggregation functions, and gloss expansions. Additionally here there is the choice of using only the gloss vector similarity term, only the sense vector similarity term, or using both, as in the original paper. Since the paper is unclear on whether the context vector should include the word currently being disambiguated, variants with both options are included.

4.3.5 SupWSD

SupWSD (Papandrea, Raganato, and Bovi 2017) is a supervised WSD system following the traditional paradigm of combining manually created features ("feature engineering") with a linear classifier, in this case, a Support Vector Machine (SVM). SupWSD is largely a reimplementation of ItMakesSense[8] (Zhong and Ng 2010), and as such uses the same feature templates and its results should be largely comparable. It was chosen over ItMakesSense since it can handle larger corpora.

All variants include the POS tag and local collocation feature templates, and the default configuration includes also the set of words in the sentence. Variants incorporating the most successful configuration of Iacobacci, Pilehvar, and Navigli (2016), exponential decay averaging of word vectors with a window size of 10, are also included for each applicable word embedding from Section 4.1.2. For each configuration incorporating word vectors, variants without the set of words in the sentence are included, denoted e.g., word2vec-s.

4.3.6 Nearest neighbour using word embeddings

Nearest neighbour using word embeddings (AWE-NN) has been used previously by Melamud, Goldberger, and Dagan (2016) as a baseline. As in Section 4.3.3, contexts are represented using an aggregation of word embeddings, but word senses are now represented by the aggregation of word embeddings of tagged instances, moving the technique from the realm of knowledge based WSD to supervised WSD. Since both tagged instances and the untagged

8. Summarised in Section 2.3.4.

context to be disambiguated are in Finnish, the constraint that word embeddings must be cross lingual is removed.

A cross lingual variant (xAWE-NN) was created based on training on SemCor (George A. Miller et al. 1994). In this variant, rather than creating an individual nearest neighbour classifier per word, during training, each example is associated with its synset. Then, during testing, for each word to be disambiguated, a list of candidate synsets is and all the associated vectors are searched for the example with the nearest neighbour vector to that of the word being disambiguated, tagging the example with the associated synset.

4.3.7 Nearest neighbour with Context2Vec

Melamud, Goldberger, and Dagan (2016) introduced Context2Vec[9] (Ctx2Vec), which models contexts using a bidirectional Long Short-Term Memory (LSTM) neural network. The model was trained as part of this thesis for 3 epochs on pre-tokenised Finnish language data originally from CommonCrawl[10]. Training was performed on surface forms, with no segmentation performed, and because of this a minimum word frequency of 100 was set so the vocabulary was small enough so that the model could be fit in memory. Language models are potentially reusable for many tasks, and this model is made available so it can be reused by others with the hope of saving the time and electrical power needed to retrain from scratch[11].

4.3.8 Nearest neighbour with BERT

BERT (Devlin et al. 2019) uses a neural network based on multiple layers of stacked attention on a task called masked language modelling[12]. A multilingual model pretrained on 104 languages made publicly available by the authors. In this model, all languages are embedded into the same space, rather than tagged or associated with a language vector. This model was used as is without further fine tuning. The model comes with a wordpiece based seg-

9. See sections 2.2.2.3 & 2.3.4.
10. See `https://commoncrawl.org/` and `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989`.
11. Available at `https://archive.org/download/ctx2vec-b100-3epoch/ctx2vec-b100-3epoch.zip`.
12. See Section 2.2.2.3.

Table 13: Baselines and calibration statistics for development corpora (%). Within each column, results are sorted in descending order.

Euro		STIFF	
Known in	99.9	Known in	100.0
Known out	82.1	Known out	77.3
1st	50.4	1st	74.3
Rand	29.7	Rand	45.7
Unambg	13.1	Unambg	36.6

mentation model which is a variant of the Byte Pair Encoding (BPE) segmentation method (Sennrich, Haddow, and Birch 2016). The segmentation is shared between all languages and is non-grammatical. It is mainly focussed on reducing the vocabulary size so as to reduce sparsity and make the model fit use less memory.

Each example was associated with the vector associated with the first segment making up the surface form of the unit to be disambiguated and classification was performed by finding the nearest neighbour (BERT-NN). As with AWE-NN, a cross lingual variant was created based on training on SemCor (xBERT-NN).

4.4 Model Selection

This section describes the model selection stage on the development data. For each of the comparison matrix tables, cells in **bold type** have the maximum score, while those with a light blue background were not found to score significantly less than the maximum scoring system. When variants that may or may not use frequency data exist, these are treated as completely different systems and model selection is run both with and without frequency data, resulting in two configurations which will both be run on the test data. Corpus calibration figures, explained in Section 4.3.1, for the development corpora used here are given in Table 13.

4.4.1 Cross lingual Lesk

Tables 14 & 15 show the results of different variants of cross lingual Lesk, with and without frequency data respectively, tested against the development data.

For the variants with frequency data, four variants are found to be not significantly different than the maximum across both training sets: pre-SIF+Concat, Average/Aggregate of Word Embeddings (AWE)+Concat, pre-SIF+fastText, pre-SIF+fastText+Expand. Of these AWE+ Concat scores maximum with on the STIFF development set while pre-SIF+fastText+Expand score maximum on EuroSense. To break this tie, we now look at the p-values with which these variants are significantly different from the maximum when tested on the other development set. AWE+Concat has p=0.09 versus the maximum variant on EuroSense, whereas pre-SIF+ fastText+Expand has p=0.08 versus either maximum variant. We select the larger p-value. This means that **Freq+AWE+Concat** is more likely to be not significantly different than the maximum EuroSense variant and therefore it is selected.

For the variants without frequency data, there are no variants in common between those variants that are not significantly different from the maximum at a p=0.05 significance level. Failing this, we opt to select the variant which is a maximum for one development set while maximally sufficing for the other development set. CATP3+Concat+Expand is maximum on STIFF while CATP3+Concat+Expand+Filter is maximum on EuroSense. The p-values versus the respective maxima in the opposing development set are 0.04 and <0.01. The variant with the larger p-value, **No Freq+CATP3+Concat+Expand** is chosen.

4.4.2 Lesk++

Table 16 show the results for different variants of Lesk++, tested against the development data. **pre-SIF+Defn+Incl. cand+Expand** is significantly better than all other techniques for both corpora. Notable here is that those variants based on the distance between the lemma vector and the context vector, denoted performs similarly to the random baseline for both corpora (see Table 13) meaning that this similarity term is not acting to perform WSD at all in this case. We can conclude that both variants are thus dragged down by the inclusion of this similarity.

Table 14: Results for variants of cross lingual Lesk with frequency data tested against development data (%). Cells in **bold type** have the maximum score, while those with a light blue background were not found to score significantly less than the maximum scoring system.

			No expand		Expand	
			No filter	Filter	No filter	Filter
EuroSense	fastText	pre-SIF	52.2	52.2	**52.4**	52.3
		AWE	51.9	52.0	51.7	52.2
		CATP3	51.7	51.8	51.7	51.7
		CATP4	51.8	51.8	51.5	51.6
	Numberbatch	pre-SIF	52.0	51.9	51.9	52.1
		AWE	51.6	51.8	51.6	51.7
		CATP3	51.6	51.7	51.9	51.9
		CATP4	51.6	51.8	51.8	51.8
	Concat	pre-SIF	52.3	52.0	51.6	51.7
		AWE	52.2	52.1	51.6	51.9
		CATP3	51.6	51.8	51.9	51.8
		CATP4	51.7	51.8	51.6	51.8
STIFF	fastText	pre-SIF	57.5	54.5	57.5	54.2
		AWE	57.7	54.2	57.8	54.2
		CATP3	57.7	54.5	**58.1**	54.5
		CATP4	57.7	54.6	57.8	54.3
	Numberbatch	pre-SIF	53.9	52.7	55.7	53.9
		AWE	55.4	53.6	55.2	54.8
		CATP3	55.1	53.9	55.1	54.2
		CATP4	55.8	54.6	55.7	54.6
	Concat	pre-SIF	57.4	54.2	57.5	54.9
		AWE	**58.1**	54.8	57.8	54.9
		CATP3	57.8	54.9	58.0	54.9
		CATP4	57.8	55.1	57.8	54.8

Table 15: Results for variants of cross lingual Lesk without frequency data tested against development data (%). Cells in **bold type** have the maximum score, while those with a light blue background were not found to score significantly less than the maximum scoring system.

			No expand		Expand	
			No filter	Filter	No filter	Filter
EuroSense	fastText	pre-SIF	35.3	34.5	41.7	40.1
		AWE	37.6	34.9	39.9	40.0
		CATP3	37.5	35.4	45.9	46.8
		CATP4	37.1	35.2	44.0	45.0
	Numberbatch	pre-SIF	33.4	33.4	35.2	35.9
		AWE	34.3	32.8	33.0	34.4
		CATP3	35.9	35.6	47.1	47.7
		CATP4	35.6	35.4	45.5	46.0
	Concat	pre-SIF	33.9	34.1	39.8	39.0
		AWE	36.7	33.1	37.0	38.3
		CATP3	36.3	35.2	47.7	**48.2**
		CATP4	36.3	35.4	45.8	46.6
STIFF	fastText	pre-SIF	39.1	36.1	43.8	42.7
		AWE	40.3	36.2	43.2	42.9
		CATP3	38.9	36.8	41.1	39.2
		CATP4	39.1	37.1	40.2	38.3
	Numberbatch	pre-SIF	36.7	36.2	39.7	38.9
		AWE	37.1	35.9	39.5	38.9
		CATP3	38.3	36.5	43.9	42.1
		CATP4	38.2	36.5	41.4	39.7
	Concat	pre-SIF	39.2	35.2	42.3	41.8
		AWE	40.5	35.9	42.9	42.3
		CATP3	39.7	37.1	**45.7**	41.8
		CATP4	40.0	37.4	43.6	40.0

Table 16: Results for variants of Lesk++ tested against development data (%). Cells in **bold type** have the maximum score, while those with a light blue background were not found to score significantly less than the maximum scoring system.

			No expand		Expand	
			Incl. cand	Excl. cand	Incl. cand	Excl. cand
EuroSense	pre-SIF	Both	32.3	32.9	35.5	34.3
		Defn	34.6	33.7	**41.3**	36.0
		Lemma	28.7	29.4	28.5	29.3
	AWE	Both	31.8	32.2	31.5	31.5
		Defn	34.4	33.9	34.5	32.6
		Lemma	28.0	28.9	27.7	28.6
STIFF	pre-SIF	Both	48.8	50.1	54.3	53.0
		Defn	49.7	49.7	**58.5**	54.7
		Lemma	49.7	50.5	49.7	50.3
	AWE	Both	50.1	50.1	52.8	53.9
		Defn	49.3	50.5	55.2	54.7
		Lemma	48.4	50.1	48.8	50.5

Table 17: Results for variants of SupWSD tested against development data (%). Cells in **bold type** have the maximum score, while those with a light blue background were not found to score significantly less than the maximum scoring system.

		default	fasttext	word2vec	fasttext-s	word2vec-s
EuroSense trained	EuroSense	62.8	**63.2**	63.1	**63.2**	63.1
	STIFF	53.9	**54.1**	53.3	53.5	53.3
STIFF trained	EuroSense	36.4	36.6	36.6	36.6	**36.6**
	STIFF	**63.4**	62.9	62.5	62.9	62.9

4.4.3 SupWSD

Table 17 shows the results for different variants of SupWSD, tested against the development data. Overall the results do not give much to choose from, given only a single test/training configuration of word2vec performs significantly worse than the best. Ultimately **fasttext** is chosen since it performs as well as the best system for three out of four train/testing configurations.

4.4.4 AWE-NN

The results of all AWE-NN variants tested against the development data are shown in Table 18. The aim is to select a variant which performs well across all testing and training set

Table 18: Results for variants of AWE-NN tested against development data (%). Cells in **bold type** have the maximum score, while those with a light blue background were not found to score significantly less than the maximum scoring system.

Training set	*Testing set*		fastText	Numberbatch	word2vec	CC2[a]	CC3[b]
EuroSense	EuroSense	pre-SIF	74.1	74.2	73.6	74.8	74.4
		AWE	74.0	74.2	73.1	74.8	73.5
		CATP3	74.0	74.9	72.9	**75.4**	73.2
		CATP4	74.0	74.7	73.1	75.4	73.3
	STIFF	pre-SIF	40.0	39.4	41.1	39.0	41.3
		AWE	40.6	40.4	41.1	**43.0**	41.3
		CATP3	40.2	41.0	42.1	42.7	42.7
		CATP4	41.1	41.1	41.9	42.9	42.5
STIFF	EuroSense	pre-SIF	31.8	31.8	32.2	32.1	32.3
		AWE	31.4	31.7	**32.5**	31.9	32.4
		CATP3	31.5	31.5	31.6	31.9	31.7
		CATP4	31.5	31.5	31.7	31.9	31.9
	STIFF	pre-SIF	83.6	81.9	**85.0**	83.8	84.6
		AWE	84.0	82.5	84.4	**85.0**	84.4
		CATP3	84.8	81.7	84.0	83.8	84.2
		CATP4	84.2	82.1	84.0	84.4	84.4

[a] Concatenation of fastText and Numberbatch

[b] Concatenation of fastText, Numberbatch and word2vec

Table 19: Baselines and calibration statistics for manually annotated corpora (%). Within each column, results are sorted in descending order.

Euro		STIFF	
Known in	85.5	Known out	84.2
Known out	75.2	Known in	81.1
1st	44.4	1st	55.8
Rand	30.8	Rand	28.4
Unambg	16.2	Unambg	15.8

combinations. Three variants using the CC2 word vectors were not found to be significantly different from the maximum across all: AWE, CATP3 and CATP4. Of these, AWE additionally achieves the maximum score when tested on STIFF data with both training sets, so **CC2+AWE** is selected to represent AWE-NN.

4.5 Results

This section presents the final results of the selected variants of the systems on the automatically annotated development and manually annotated test sections of both the EuroSense and STIFF corpora. Corpus calibration figures, explained in Section 4.3.1, for the manually annotated corpora used here are given in Table 19, while the final WSD results are given in Table 20. The rankings of the systems vary across the different evaluation corpora and this permutation structure is illustrated in Figure 18.

Firstly we note that the test set size was not big enough to differentiate the top systems. For EuroSense and STIFF we estimate how many sense tagged instances would be required to find a significant difference between the top system and the next two systems scoring at least percentage point less. This was done by repeating the bootstrapping procedure described in Section 4.2.3 but modifying Algorithm 1 to vary the size of the schedule matrix t from the actual sample size n to an extrapolated sample size. The resulting plot is shown in Figure 19. Note that this analysis is an extrapolation and only says that we would reach a p-value of 0.05 if we annotated more data and found that the score distribution remained the same on this new annotated data. It is possible that the new annotated data could change ranking of the top systems. Using this plot, and adding a small safety margin, to obtain significance

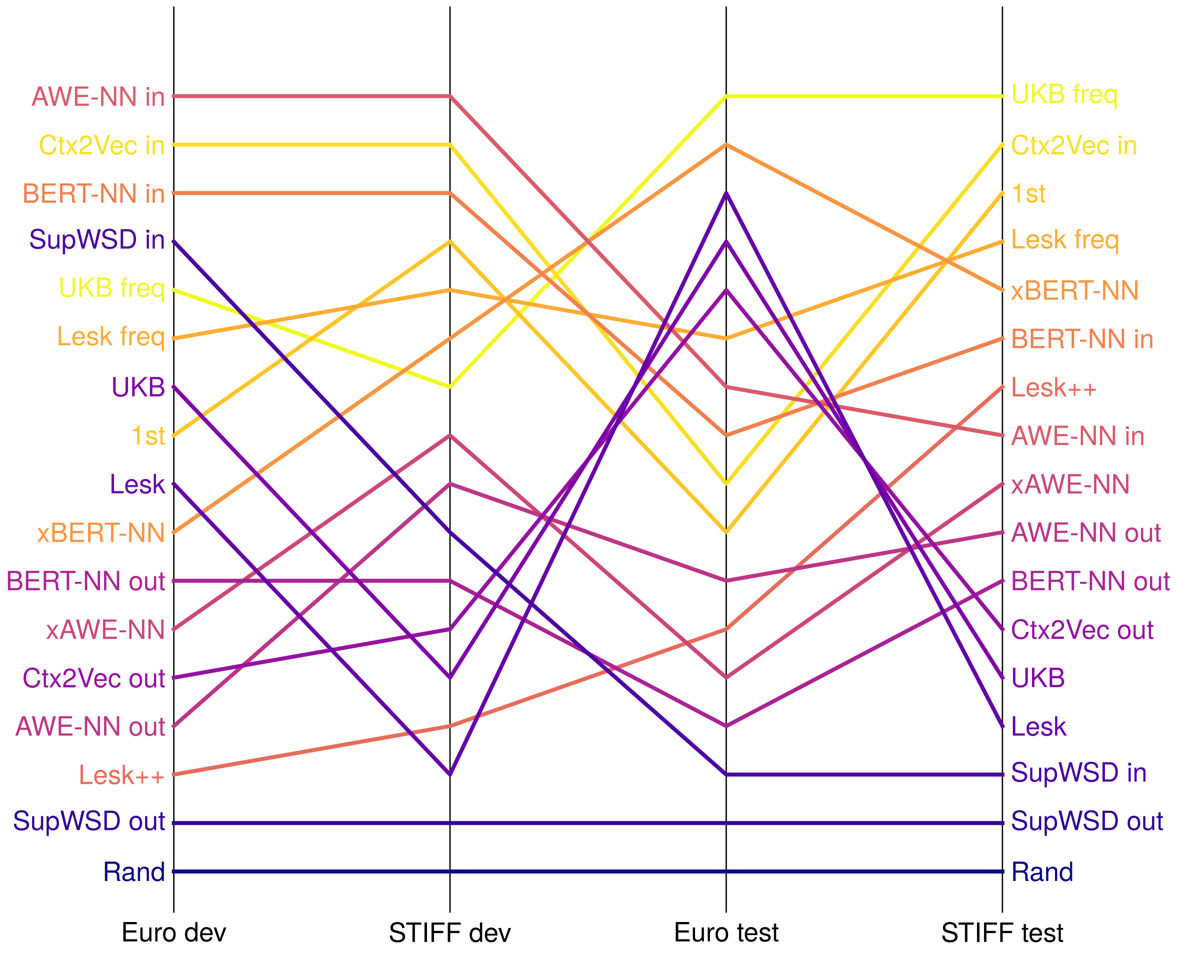


Figure 18: Parallel coordinate plot to emphasise the permutation structure of the ranks of the systems on the different test corpora. See also Table 20.

Table 20: Results for final variants of WSD systems tested against development and manually annotated sections of both corpora (%). Within each column, systems are sorted in descending order of their F_1-score. The permutation structure is emphasised in the accompanying parallel coordinate plot in Figure 18. Systems with the same letter next to their score are not found to have a significant difference between their results, as described on page 80. Manually annotated corpora are indicated as **bold** to emphasise their reliability over the development corpora. Results in *italics* are the development corpus result of a system which has had model selection performed on that same development corpus.

Euro dev		STIFF dev		**Euro test**		**STIFF test**	
AWE-NN in	*74.9_{a}*	*AWE-NN in*	*86.1_{a}*	UKB freq	$51.5_{a,b}$	UKB freq	57.9_{a}
Ctx2Vec in	67.6_{b}	Ctx2Vec in	81.1_{b}	xBERT-NN	$51.3_{b,c}$	Ctx2Vec in	$56.8_{a,b}$
BERT-NN in	67.3_{b}	BERT-NN in	79.8_{b}	xLesk	$49.6_{b,c,d,e}$	1st	$55.8_{a,b}$
SupWSD in	*63.2_{c}*	1st	74.3_{c}	UKB	$48.9_{a,b,c,d,e,f,g}$	xLesk freq	$55.8_{a,b}$
UKB freq	54.4_{d}	*xLesk freq*	*73.3_{c}*	Ctx2Vec out	$47.9_{a,b,c,e,g}$	xBERT-NN	$55.8_{a,b}$
xLesk freq	*52.2_{e}*	xBERT-NN	68.4_{d}	xLesk freq	$47.0_{c,d,e,f,g,h}$	BERT-NN in	$55.8_{a,b}$
UKB	51.7_{e}	UKB freq	65.3_{e}	AWE-NN in	$45.3_{a,b,c,d,e,f,g,h}$	Lesk++	$54.7_{a,b}$
1st	50.4_{f}	xAWE-NN	$64.0_{e,f}$	BERT-NN in	$45.3_{a,b,c,d,e,f,g,h}$	AWE-NN in	$54.7_{a,b}$
xLesk	*47.7_{g}*	*AWE-NN out*	*64.0_{e}*	Ctx2Vec in	$45.3_{a,b,c,d,e,f,g,h}$	xAWE-NN	$50.5_{a,b,c}$
xBERT-NN	44.1_{h}	*SupWSD in*	*$62.9_{e,f,g}$*	1st	$44.4_{c,d,e,f,g,h}$	AWE-NN out	$50.5_{a,b,c}$
BERT-NN out	44.1_{h}	BERT-NN out	$62.3_{e,f,g}$	AWE-NN out	$44.4_{c,d,e,f,g,h}$	BERT-NN out	$50.5_{b,c}$
xAWE-NN	43.6_{h}	Ctx2Vec out	$61.3_{f,g,h}$	Lesk++	$42.7_{a,f,g,h}$	Ctx2Vec out	$50.5_{b,c}$
Ctx2Vec out	43.6_{h}	UKB	$60.2_{g,h}$	xAWE-NN	$42.7_{d,e,f,g,h}$	UKB	$50.3_{b,c}$
AWE-NN out	*43.5_{h}*	*Lesk++*	*58.5_{h}*	BERT-NN out	$42.7_{d,f,h}$	xLesk	*$45.3_{c,d}$*
Lesk++	41.3_{i}	*xLesk*	*57.9_{h}*	SupWSD in	$41.9_{d,e,f,g,h}$	SupWSD in	$40.0_{d,e}$
SupWSD out	*36.6_{j}*	*SupWSD out*	*54.1_{i}*	SupWSD out	38.5_{h}	SupWSD out	35.8_{e}
Rand	29.7_{k}	Rand	45.7_{j}	Rand	30.8_{i}	Rand	28.4_{f}

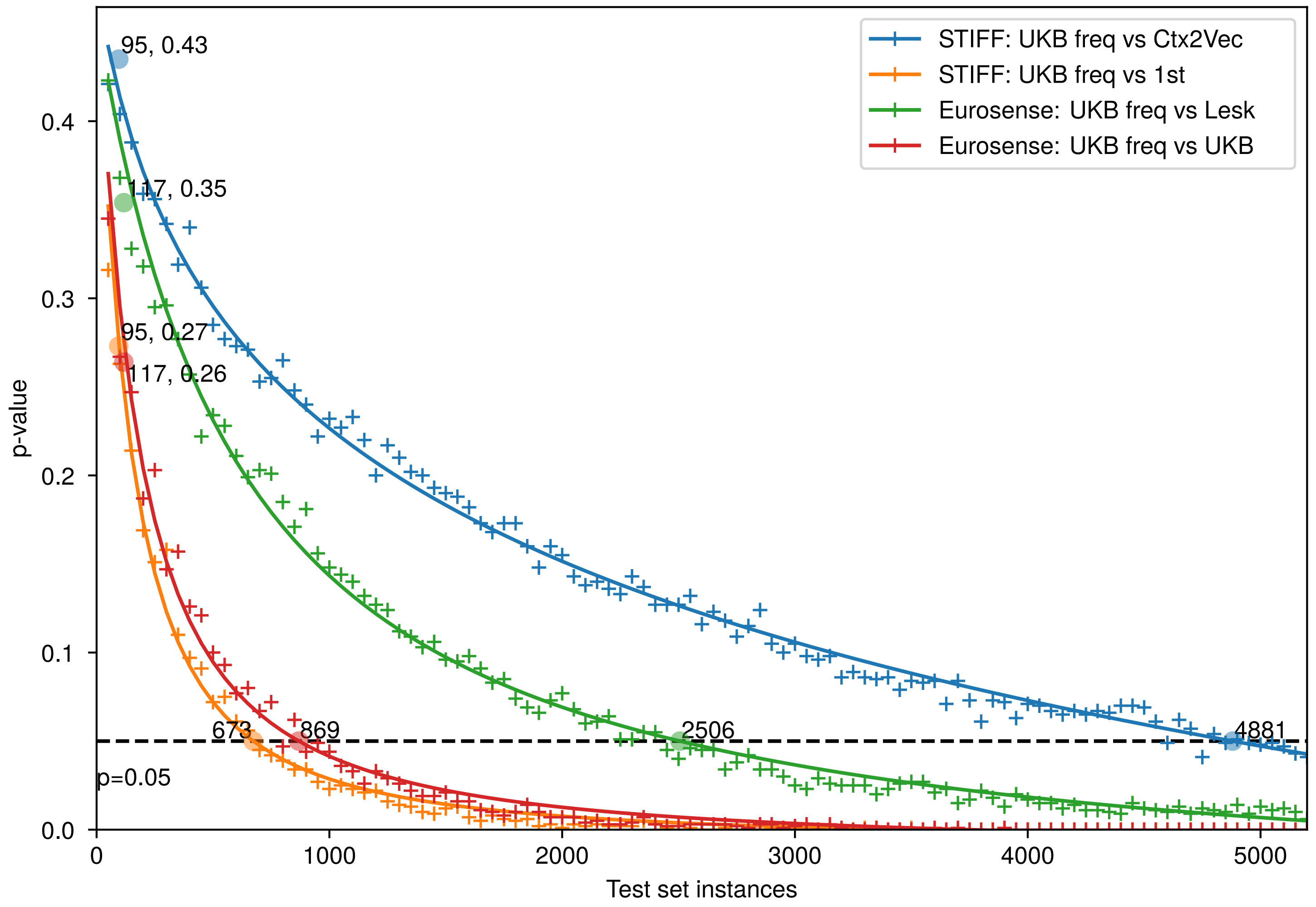


Figure 19: Graph showing an extrapolation estimating the p-values for a significant level of difference between pairs of top performing systems decreasing with increasing sample size. Lines of the form $\frac{a}{((x+b)^d)+c}$ have been fitted.

between UKB freq and 1st on STIFF and UKB freq and xLesk on EuroSense we would need to obtain in total 1000 sense tagged instances for each of STIFF and EuroSense, or about 90 more hours/2 full time weeks of annotation. To obtain a difference between UKB freq and Ctx2Vec for STIFF and UKB freq and UKB for EuroSense would require in total 5000 instances, or about 490 more hours/12 full time weeks of annotation.

In terms of statistically significant results, UKB freq manages to clear the 1st sense baseline on the EuroSense test data — the only system to do so on either test section. All systems managed to clear the random baseline on all sections. SupWSD performed worse than the 1st sense baseline on both test sections, except for the in domain variant on EuroSense. Acknowledging that more manually annotated data is required to obtain statistical significance, the rest of this section relaxes adherence to the p=0.05 level.

Now we examine the effect on in and out of domain training data and evaluating on automatically versus manually annotated data. It is evident that the in domain supervised systems are better able to fit the systematic errors from the automatically annotated development sets than the manually annotated test sets, while the out of domain and cross lingual trained versions of the supervised systems fall behind or perform similarity to non-supervised systems. When testing on the manually annotated data the effect of domain becomes much less pronounced, with cross lingual BERT systems beating any in domain systems and the margin between in and out of domain variants of the same systems being much smaller. The fact that in domain AWE-NN is the best system on the development data but is nowhere near on the test data indicates that this system has overfitted the data, possibly due to the model selection stage.

Secondly, we turn to what degree systems are able to beat the 1st sense baseline. For both development and test sections, many more systems clear the baseline for EuroSense than for STIFF. In fact, for the test section of STIFF, the only system to actually beat the 1st sense baseline, UKB freq, actually incorporates it. The likely explanation for this difference stems from the choice of sentence length contexts. Being derived from parliamentary transcripts, EuroSense contains lengthy, formal language and thus quite long sentences/contexts, whereas STIFF is derived from subtitles and thus contains quite short sentences/contexts (often sentence fragments or even single words). Figure 20 presents a histogram showing frequently different context sizes appear for different instances. Since all systems apart from the 1st sense baseline rely on the context to disambiguate, STIFF is quite strongly disadvantaged here. Another reason may be that both the development and test sections of EuroSense have sense frequencies distributions less in line with the data the 1st sense baseline is ultimately derived from when compared to STIFF. Referring to information about the test sections of EuroSense and STIFF presented in Table 19, the unambiguous, random sense and known tokens are roughly comparable, all being within 5 percentage points, but the 1st sense baseline is 10 percentage points more for STIFF. This may be because the EuroSense data represents are narrower domain than STIFF.

Now we turn to comparison with previous results, in particular, we compare the results of the English language evaluation of Raganato, Camacho-Collados, and Navigli (2017) on all concatenated test sets (Raganato, Camacho-Collados, and Navigli 2017, Table 4) to the results on the manually annotated test corpora sections. In Raganato, Camacho-Collados, and Navigli (2017) evaluation all configurations of ItMakesSense, as well as Context2Vec, outperform the 1st sense baseline, while UKB falls short of it. Noting that SupWSD and ItMakesSense contain identical implementation approaches to WSD and thus should be comparable, how can we explain this discrepancy? Firstly, we reiterate from Section 4.3.2 that Raganato, Camacho-Collados, and Navigli (2017) presents a version of UKB which does not incorporate any frequency information. Even so, UKB without frequency information is quite competitive for the EuroSense test section. This may be due to EuroSense's contexts being larger than the test corpora in that evaluation. Indeed it was noted by Agirre, López de Lacalle, and Soroa (2018) that UKB needs quite a bit of context to perform well, to the point where the context was extended beyond a single sentence in their experiments.

Less clear is why for example, SupWSD and Context2Vec (Melamud, Goldberger, and Dagan 2016)[13], which were shown to perform well for English fare so poorly here. One possibility is that there are negative effects due to unique features of the Finnish language or due to the way FinnWordNet is created. We could also bring the manually annotated corpus created in Section 3.5 into doubt. However, the most seemingly likely explanation is that training solely on automatically induced corpora leads to poor performance for these systems. We need only

13. The result for Context2Vec on the concatenated dataset of Raganato, Camacho-Collados, and Navigli (2017) is available at `http://nlpprogress.com/english/word_sense_disambiguation.html`. It was communicated directly by the author.

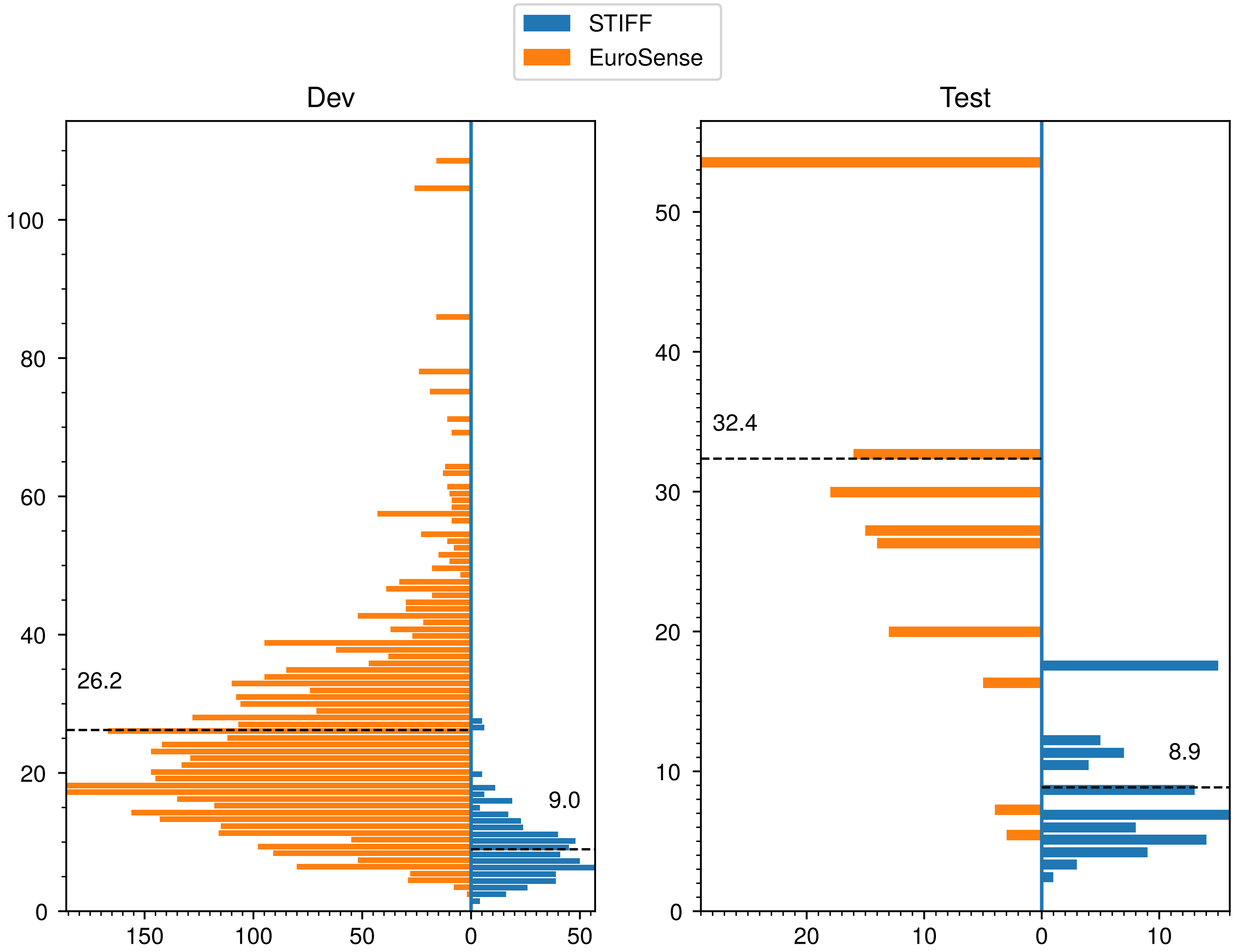


Figure 20: Histograms comparing the context lengths in tokens of the development and test section of STIFF and EuroSense. The mean is indicated with a dashed line.

refer to Figure 13 to see that both training corpora have quite low precision and recall, and in particular, the figures obtained for EuroSense are much lower in the evaluation of its Finnish section presented here than the other languages evaluated in their paper. In the experiments of Bovi et al. (2017) all supervised systems were trained on the concatenation of manually annotated corpora *and* automatic corpora which might mitigate the worst biasing effects of using the automatic corpora. The good results for cross lingual BERT support this, showing that in this case, using a manually annotated corpus from another language outperforms using an automatically induced one from the same language.

In the case of Context2Vec, it is possible a different training setup would result in a better language model and therefore better results here. Namely cutting off words with a frequency of less than 100 is clearly quite undesirable, and particularly with an agglutinative language like Finnish performing segmentation, either grammatically motivated or purely aimed at reducing vocabulary size like BPE could well improve the result since it would address the sparsity caused by using surface forms.

Another result which we were not able to replicate here is Lesk++ (Oele and Noord 2017). The main difficulty seems to be that the cosine similarity between the context vector and lemma vector. This was a reimplementation, so one possibility is bugs in the implementation, however, given the other term appears to work correctly and the implementation was checked over several times this seems unlikely. It could also result from the fact that the exact process the word vectors were made was quite different, using Numberbatch vectors as a starting point rather than word2vec vectors, resulting in the vector space which contains surface lemmas, synsets (synset, lemma) pairs not working as intended. However, in Section 5.3 it is used as the basis of an approach to word sense clustering which is competitive with other approaches and so it appears that this embedding is not completely broken.

The implementations of the techniques reimplemented for this evaluation and the scripts and configuration files for the adapted open source systems are publicly available under the Apache v2 license. To ease replicability further, the entire evaluation framework, including all the requirements, WSD systems and lexical resources are made available as a Docker image[14].

14. Available at `https://github.com/frankier/finn-wsd-eval`.

5 Creating an aligned Finnish lexical resource

This chapter introduces a lexical resource made by combining automatically extracted Wiktionary definitions, extracted in Section 5.1, with FinnWordNet (FiWN). There are two advantages to basing TheWhatNow?! on a resource like this:

- It increases the coverage of Finnish vocabulary;
- It allows prioritisation of Wiktionary definitions where possible. These tend to be of higher quality. The reason for this is:
  - They are written for humans first, whereas FiWN is arranged around synsets first, resulting in overly fine grained senses;
  - They are written for the particular Finnish word, rather than for a synset, and may, therefore, be more precise;
  - They can contain other information not available in FiWN, such as grammatical information about common inflections of associated words, explored in Section 5.2.

On the other hand, as we have seen, FiWN has advantages too. Being based on Princeton WordNet (PWN), it is linked to other languages, a fact used to induce a sense tagged corpus in Chapter 3. Being somewhat of a standard among Lexical Knowledge Bases (LKBs), means that is it compatible with the greatest number of Word Sense Disambiguation (WSD) techniques, a fact used in Chapter 4. Therefore we would like to combine Wiktionary definitions and FiWN to gain the benefits of both resources. The approach used here is to jointly align and cluster senses, and this is done in Section 5.3. Another important issue is that of lexical items more complex than simply a single lemma, including for example Multi-Word Expressions (MWEs) in the LKB which need to be recognised and extracted from free text as part of the implementation of the TheWhatNow?!, and this is addressed in Section 5.2.

5.1 Scraping Wiktionary

Wiktionary definitions are typically edited as a mixture of unstructured text alongside semi-structured markup written in with the MediaWiki markup language. The MediaWiki markup language allows for semi-structured data entry using templates. However, there is very little structure of a Wiktionary page which is strictly enforced by the MediaWiki software. Instead, what consistent structure there is created by convention and moderation, with some editors dedicating much of their energy to 'cleaning up' the edits of others. Typically each Wiktionary page can one or more language headings. Within each language heading, there may be one or more etymologies, and within each etymology, there may be one or more Part Of Speech (POS) headings. Most commonly, however, a word has only a single etymology, and in this case, POS headings sit directly below language headings.

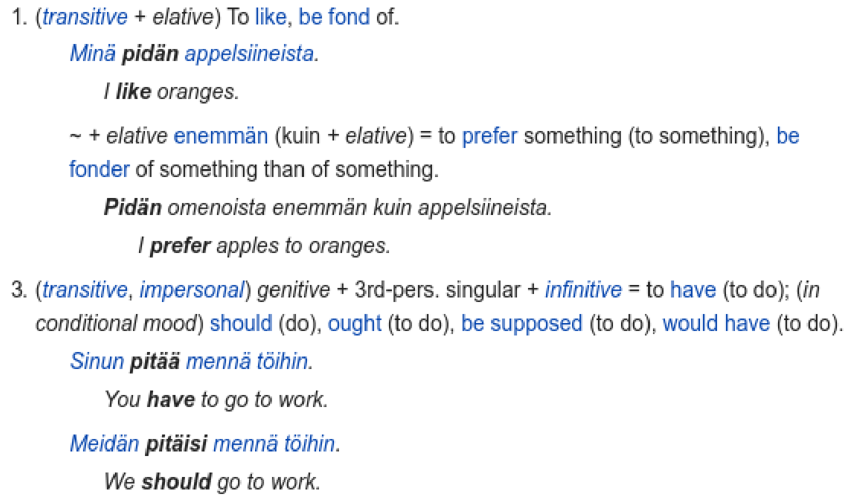
Under each POS heading, there are one or more definitions. These are usually formatted as a nested list, possibly in combination with templates. This nested list may also include examples, grammar notes and collocations. A typical example is given in Figure 21. Notable is that a definition may contain either subsenses or examples underneath it.

The scraping software, created as part of this thesis, reads Wiktionary dumps. The dumps are formatted as eXtensible Markup Language (XML), read using the *mwxml* library[1], while the MediaWiki markup within is parsed using the *mwparserfromhell* library[2]. The choice to work on the raw MediaWiki markup rather than the resulting HyperText Markup Language (HTML) on the one hand makes things easier, since it gives access to templates, which can contain structured information in their arguments. On the other hand, it can mean that more variants need to be handled, since in some cases, inexperienced editors do not use templates, but instead use MediaWiki markup equivalent to the output of the template. In addition in some cases, there are multiple templates with the same purpose.

The main difficulty for the parser is the problem of determining whether some text is part of the definition text itself, part of a grammar note or collocation for a definition (such as + *elative*, or kuin + *elative*), part of a note not concerning grammar (e.g., a tag such as "dated" or domain such as "aeronautics"), or part of a usage example. When the text is part of a

1. Obtained from `https://github.com/mediawiki-utilities/python-mwxml`.
2. Obtained from `https://github.com/earwig/mwparserfromhell`.

```
# {{lb|fi|transitive|_|+ elative}} To [[like]],
 ↪  [[be]] [[fond]] of.
#: ''[[minä|Minä]] '''pidän'''
 ↪  [[appelsiini|appelsiineista]].''
#:: ''I '''like''' oranges.''
#: ~ ''+ elative'' [[enemmän]] (kuin ''+
 ↪  elative'') = to [[prefer]] something (to
 ↪  something), [[be]] [[fonder]] of something
 ↪  than of something.
#:: '''''Pidän''' omenoista enemmän kuin
 ↪  appelsiineista.''
#::: ''I '''prefer''' apples to oranges.''
# {{lb|fi|transitive|impersonal}} ''genitive'' +
 ↪  3rd-pers. singular + ''[[infinitive]]'' = to
 ↪  [[have]] (to do); (''in conditional mood'')
 ↪  [[should]] (do), [[ought]] (to do), [[be]]
 ↪  [[suppose]]d (to do), [[would]] [[have]] (to
 ↪  do).
#: ''[[sinä|Sinun]] '''pitää''' [[mennä]]
 ↪  [[työ|töihin]].''
#:: ''You '''have''' to go to work.''
#: ''[[meidän|Meidän]] '''pitäisi''' [[mennä]]
 ↪  [[työ|töihin]].''
#:: ''We '''should''' go to work.''
```



1. (*transitive* + *elative*) To like, be fond of.

   *Minä **pidän** appelsiineista.*

   *I **like** oranges.*

   ~ + *elative* enemmän (kuin + *elative*) = to prefer something (to something), be fonder of something than of something.

   ***Pidän** omenoista enemmän kuin appelsiineista.*

   *I **prefer** apples to oranges.*

3. (*transitive*, *impersonal*) *genitive* + 3rd-pers. singular + *infinitive* = to have (to do); (*in conditional mood*) should (do), ought (to do), be supposed (to do), would have (to do).

   *Sinun **pitää** mennä töihin.*

   *You **have** to go to work.*

   *Meidän **pitäisi** mennä töihin.*

   *We **should** go to work.*

Figure 21: Two definitions of pitää from Wiktionary. On the left: as written using MediaWiki markup. On the right: as rendered in a web browser.

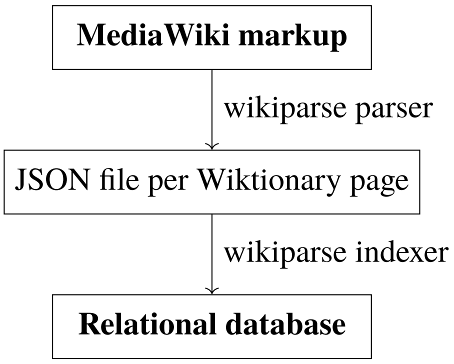


Figure 22: The wikiparse pipeline.

usage example, there is further the question of whether it is a Finnish language usage, which all Finnish word examples should have, or an English language translation, which many but not all Finnish usage examples have.

In some cases, this is clear, due to certain template tags being used, such as the *defn* template tag for definition text. In other cases, the presence of certain words or symbols, such as *elative*, or + are used. When there is a problem, such as no heuristic can be applied, then the part of the definition tree being processed is discarded.

5.1.1 Structured data format

There are two levels of structured data format output by wikiparse. A JavaScript Object Notation (JSON) output and a relational database output. The parser program is structured as a cascade of nested iterators, forming a coroutine structure, where inner levels yield more detailed information, while outer levels have access to more context to integrate the results of the inner level. This naturally leads to producing a nested structure as a result. The JSON format is the outermost level of this structure. The relational database is created primarily for the purpose of efficient access, acting in principle as a series of indexes and views on top of the JSON data. The pipeline is shown in Figure 22.

An example of the JSON format for the senses of pitää, previously shown in MediaWiki format in Figure 21, is shown in Figure 23. Many empty or null fields are omitted, as are subsenses. Within the `defns` key, definitions are structured according to the tree structure of the Wiktionary page. The definition text itself is extracted into several forms: the raw

MediaWiki markup `raw_defn`, saved for further parsing; a version with only basic MediaWiki markup and templates either expanded or removed, `cleaned_defn`; and a plain text version, `stripped_defn`. Usage examples are placed under definitions. The assoc entry gives information, which is used for creating extraction schemas in Section 5.2. Within the heads key, headword level information is extracted: etymology heading grouped etymologies and relations.

The schema of the relational database is shown in Figure 24. Fields labelled extra are populated the JSON fragment the row is derived from, to allow access to all data from the JSON format. The schema has some degree of denormalisation since different tables have been created specifically to support different users. The `headword` and `word_sense` tables are the main tables, and are used to obtain the actual definition text for display by TheWhatNow?! in Chapter 6 as well as for definition clustering in Section 5.3. While the `headword` table is sufficient for looking up most definitions, for some complex lexical items such as MWEs, special purpose indexing is need, and this is constructed in Section 5.2. The `inflection_of` table is primarily used for determining whether senses are lemma senses, for example, to make sure the manually clustered senses of Section 5.3.3 are lemmas. The `relation` table encodes relations such as misspellings and dialectical forms and is used by TheWhatNow?! to pull in extra relevant definitions. Finally, the `etymology`, `derivation` and `derivation_seg` tables contain headword level etymology and word formation information, including compounding, word derivation and inflection. These are used as part of the analytical segmenter of Section 6.2.

5.1.2 Evaluation

Due to the variation in formatting, it is likely that we cannot entirely accurately scrape all information from Wiktionary. Table 21 shows the possible outcomes for some datum to be extracted. In this setting, we now abandon any notion of a gold standard. Since we cannot refer to any best possible system, we cannot successfully determine the difference between True Positives (TPs) and False Positives (FPs), only the number of data, corresponding to all positives, the actual system returns (TP + FP). In some cases, we can identify all cases in which we might extract some information but failed to, i.e., the False Negatives (FNs). With

```
{"defns": {"Verb": [
   {"raw_defn": " {{lb|fi|transitive|_|+ elative}} To
    ↪  [[like]], [[be]] [[fond]] of.\n",
    "cleaned_defn": "  To [[like]], [[be]] [[fond]] of.\n",
    "stripped_defn": "To like, be fond of.",
    "assoc": {
      "verb": ["transitive"],
      "obj": ["elative"]
    },
    "bi_examples": [{
      "fi": ["Minä pidän appelsiineista."],
      "en": ["I like oranges."]
     }],
    "subsenses": [{...}]},
   {"raw_defn": " {{lb|fi|transitive|impersonal}}
    ↪  ''genitive'' + 3rd-pers. singular + ''[[infinitive]]''
    ↪  = to [[have]] (to do)",
    "cleaned_defn": " to [[have]] (to do)",
    "stripped_defn": "to have (to do)",
    "assoc": {
      "subj": ["genitive"],
      "verb": ["transitive", "impersonal", "3rd-pers.
       ↪  singular", "infinitive"],
    },
    "bi_examples": [{
      "fi": ["Sinun pitää mennä töihin."],
      "en": ["You have to go to work."]
    }],
    "subsenses": [{...}]}]},
 "heads": [{
   "ety_idx": null,
   "poses": ["Verb"],
   "tag": "etymology-heading",
 }]}
```

Figure 23: An example of wikiparse's JSON format.

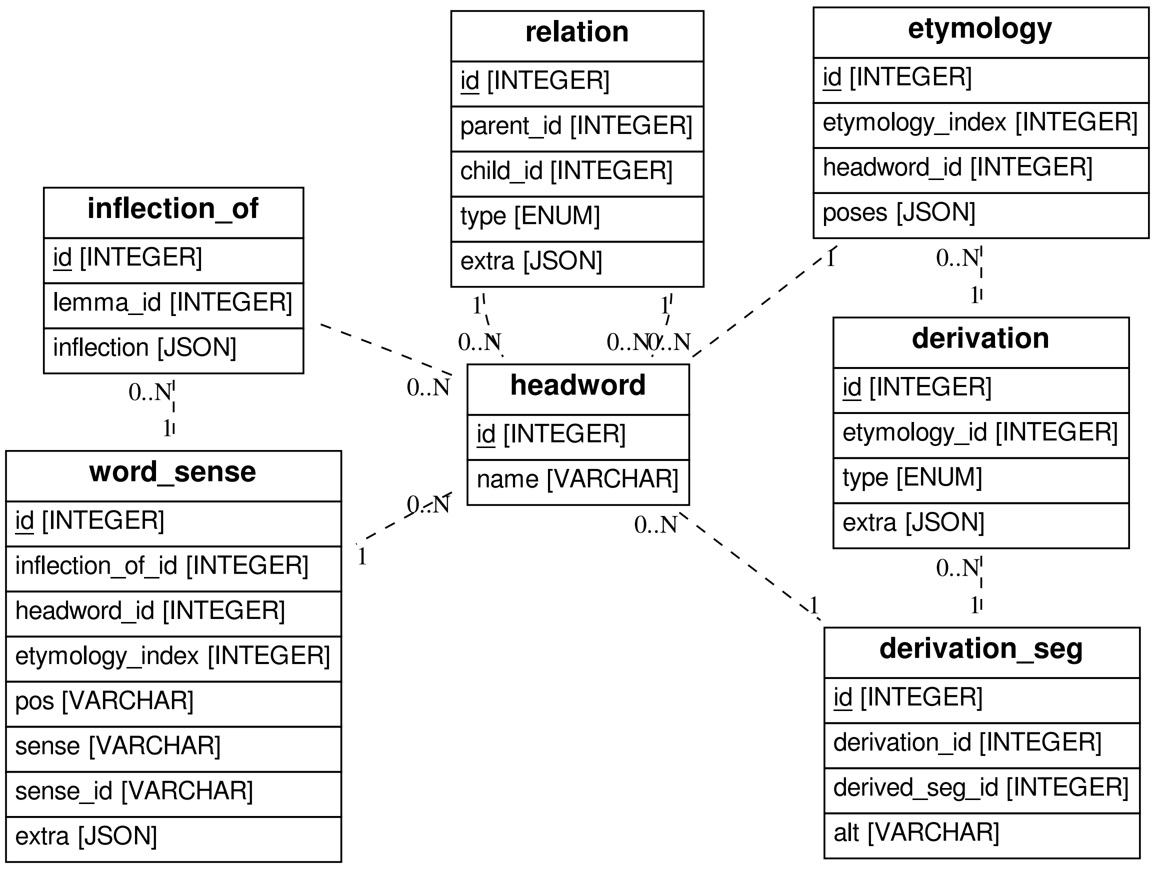


Figure 24: The schema of wikiparse's relational database format as an entity relation diagram.

Table 21: Table of possible outcomes for Wiktionary parser.

		Best possible system	
		Positive	Negative
Actual system	Positive	TP	FP
	Negative	FN	TN

the information we can calculate metric is known as coverage:

$$coverage = \frac{TP + FP}{TP + FP + FN}$$

Because coverage doesn't take into account FPs, it should be noted as such and the development of the system undertaken in such a way as to minimise them. During the development of the project, FPs must be checked for manually and considered as bugs. To this end, some of the most popular words were manually checked for incorrect extracted information (FPs) during the development of the scraper. Based on this an automated test suite was created so that there would not be regressions for these particular classes of FPs. When the resulting data is integrated into an end system, further FPs can be fed back. In summary, although overall recall and precision are not measured, the development process is directed towards building a high precision system, if this is done successfully, the coverage measurement can then be used to approximate recall since:

$$\begin{aligned} recall &= \frac{TP}{TP + FN} \\ &\approx \frac{TP + FP}{TP + FP + FN} \quad \text{(when FP is small)} \end{aligned}$$

Using this approach makes coverage a useful measure for consulting during the development of the scraper. However, without reference to a gold standard, there is no real possibility of quantitative comparison between alternative approaches, due to the issue of the FNs.

Ideally, we would convert FPs into FNs, usually at the cost of also converting TPs into FNs, i.e., trading recall for precision, since we can find out more information about FNs. In particular, when the system encounters something that it does not know how to process, such as a template it doesn't recognise, or some text it can not determine whether it is an example

Table 22: Table of problems from running Wiktionary parser (%).

Problem	Importance
unknown structure	43.0
unknown structure / unknown-template-ety	34.7
unknown structure / unknown-template-ety / m	24.3
unknown structure / unknown-template-ety / cog	13.4
unknown structure / unknown-template-ety / inh	12.0
unknown structure / unknown-template-ety / der	9.5
unknown structure / not-ux-lb	7.0
unknown structure / expect-only	5.3
unknown structure / not-ux-lb / n-g	4.4
unknown structure / mwe-ety	4.2
unknown structure / expect-only / fi examples / unk examples / senses	4.1
defns empty	3.9
unknown structure / no-grammar-=	2.4
unknown structure / not-ux-lb / lb|fi	2.2
unknown structure / non-fin-assoc	1.6
unknown structure / unknown-template-ety / bor	1.2
unknown structure / not-ux-lb / q	1.1
unknown structure / expect-only / bi examples / senses	0.9
unknown structure / unknown-template-ety / fi-form of	0.7
unknown structure / not-ux-lb / uxi	0.6

or a definition then details of this event can be logged. Later, the events can be grouped and aggregated so as to find which parts of the system should be focussed on next to maximally increase coverage.

5.1.3 Results

The dump used in this section is from the 6th of April 2019. In this dump, there are 158 227 pages with a Finnish section. Some data is extracted from 97.4% of pages, whereas data is extracted without encountering any problems for 89.8% of pages. Note again that as explained in Section 5.1.2, this figure is merely the proportion of pages where the parser has not encountered a known error — it does not mean that there are definitely no problems with the data extracted from these pages.

Those problems that do occur are broken down in Table 22. The importance measure is obtained by weighting the word for which the problems occurs by its frequency:

$$\text{importance} = \sum_{w \in p} \text{freq}(w)$$

.

The resulting measure is presented as a percentage, however, since they are not mutually exclusive, they do not sum to 100%. The word frequencies used are those of Speer et al. (2018)[3]. Note that smoothing is used, so out of vocabulary words are assigned a small, non-zero frequency.

The problems are arranged into a tree like structure to allow identification of the largest problems as a coarse and finer grained level. The most common type of problem is some kind of `unknown structure`. In particular, the high rate of occurrence of `unknown-template-ety` results shows that templates in the etymology are not yet all handled yet. While `not-ux-lb` errors are generally caused by unknown templates within a sense definition.

5.2 MWEs and schemas

Finnish has a rich morphology[4]. Substantives are declined for case and number and verbs are conjugated for person, tense and voice. Finnish word formation is also rich[5]. It includes a number of highly productive derivational morphemes, including many deverbal morphemes which is characteristic of the language. Compounding also plays a major role in Finnish word formation, with many of the compounds being semantically transparent. Compared to a language like English for example, Finnish word formation may be more predictable since it draws root words from a smaller pool and has a richer set of primitives to combine them. Finnish also has a number of enclitic particles such as the question forming "-ko". Finally, it has MWEs such as idioms. In Finnish these may take the form of schemas, defined here

3. Obtained from `https://github.com/LuminosoInsight/wordfreq`.
4. See for example Karlsson (2015) and Section 2.1.
5. See for example Hyvärinen (2019).

as MWEs with gaps such as “pitää ___-sta” which could occur in a form such as “pidän voileipäkakusta”.

From this point forward, we refer to any item which can be given a definition, including lemmas, individual morphemes, combinations of morphemes and MWE expressions including schemas as *headwords*.

5.2.1 Obtaining schemas

Word definitions are obtained two sources: FiWN (Lindén and Carlson 2010) and the Wiktionary definitions from Section 5.1. Both Wiktionary and FiWN contain data which can be used to create schemas. On Wiktionary, the data is included within the text of a definition, for example, the headword pitää has the entry “(transitive + elative) To like, be fond of”. In this case, the schema headword “pitää ___-sta” is extracted and associated with this definition. FiWN has separate headwords for schemas. They are marked using abbreviated forms of pronouns. For example in the headword “pitää_kiinni_jstak” jstak is short for jostakin, allowing the schema “pitää kiinni ___-sta” to be extracted.

5.2.2 Headword extraction

In order to extract headwords from free text, two approaches were taken: one for MWEs, schemas and inflected headwords and one for single word lemma headwords including headwords which are part of a word, e.g., a constituent words of a compound. The former is described here, while the latter is performed as part of the segmentation process described in Section 6.2.

To find inflected headwords and MWEs including schemas, each headword is first preprocessed into a list of lemmas and required features. For an inflection this list will consist of a single item, for example, vahingossa (accidentally) will become the lemma vahinko and the feature case=inessive. An MWE will be much the same as an inflection, but the list will have multiple entries. For a schema, one or more entries have a gap in place of the lemma, this gap acts as a wildcard able to match any lemma. Next, of the words with a lemma, the least common according to a frequency list of Finnish lemmas, is chosen as the key lemma.

The lemmatisation process is based upon Omorfi (Pirinen 2015b). At run time Omorfi analyses Finnish words using a Finite-State Transducer (FST): an automaton which acts as a two way mapping between surface forms and analyses. The source data for Omorfi is a list of lemmas and their paradigms, a category which determines how different forms of the lemma will be realised in the surface form. The source data is obtained from historical versions of Finnish and English Wiktionary, the new Finnish word list produced by the Institute for the Languages of Finland, and Joukahainen which is a dictionary created for usage by the open source Finnish spell checker Voikko.[6]

Unlike Omorfi, the objective here is to produce only lemmas that span the whole token. This means that only inflections should be removed, not derivational morphemes, and also that non-final constituent words in compounds should be left as is. To achieve this, first, an Omorfi analysis is performed. Next, it is stripped back as far as, but no further than, the last derivation boundary. Finally, the ending of a lemma analysis is appended and an Omorfi generation is performed upon the resulting analysis. The features are taken from that part of the analysis which is removed. This process is performed by the *finntk* Python library[7].

There are then two possible procedures for extracting the headword. The first works on tokenised text and relies on the headword appearing in order in the text. This method lemmatises each token in the text using the same lemmatisation procedure as above. If there is a matching key, the features of the key are checked to make sure they are present in the token being matched. Next, the same check is performed towards the left and right of the match point. For wildcards lemmas, no lemma matching is performed, only feature matching. Wildcards can also match against more than one token.

The second procedure operates on dependency trees, obtained using the Turku Finnish dependency parser pipeline[8]. After finding a key match in the same manner as the token based method, the process continues to match any other part of the headword against any neighbour, extending the neighbourhood in the process until the whole headword is matched or it

6. For further analysis of Omorfi, see also Robertson (2016).
7. Available at `https://github.com/frankier/finntk`.
8. Obtained from `https://github.com/TurkuNLP/Finnish-dep-parser`.

is impossible to proceed. Special care is taken of wildcards so that they can only match across multiple tokens which are contiguous within the dependency tree.

The performance of this technique relies upon the matching lemmas occurring relatively irregularly within normal texts, meaning the full matching procedure does not occur in the common case. This approach is related to one from the string matching literature, where for example Tarhio, Holub, and Giaquinta (2017) show that given an efficient procedure to scan for a single character, searching for a literal string by searching first for its rarest character can outperform more complex methods. The headword indexing and extraction methods are part of the lextract Python library[9].

5.3 Sense clustering

FiWN is modelled after WordNet, and as such has very fine grained sense distinctions. This results in potentially overwhelming the learner with too much information. Furthermore, some Wiktionary senses are likely to essentially duplicate FiWN. Thus, to try and display the most relevant information first in TheWhatNow?! in Chapter 6, similar definitions should be clustered together and only the best definition displayed by default.

The section compares a few approaches. A significant piece of work with a similar objective for English is that of Snow et al. (2007). They use a supervised approach to select features from many different sources of information, including for example resources which are only freely available for non-commercial use such as that of Navigli (2006) who automatically align WordNet synsets with word definitions from Oxford dictionary (which are more coarse grained). The approach of Snow et al. (2007) uses hierarchical clustering so that any number of clusters can be chosen. In comparison to that work, here a completely unsupervised approach is taken, meaning no feature selection is used. Secondly, some experiments also include Wiktionary senses in the clustering alongside WordNet senses. Thirdly, the resources used here are all liberally licensed. Finally, the perspective here is that there is some best clustering with a fixed number of clusters which allows the use of non-hierarchical clustering algorithms.

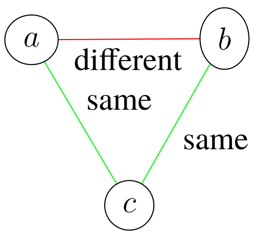
9. Available at `https://github.com/frankier/lextract`.

5.3.1 Partitions and same-different graphs

We now introduce the notion of a *same-different graph* as an interpretation of a partition as a device used in sections 5.3.2 and 5.3.4. Given a set S, its partition P, and a function from a member of S to the member in the partition P which contains S, $c\colon S \to P$, its same-different graph is a complete graph with S as its set of nodes. Each edge has a type given by a function $t\colon S \times S \to \{\text{same}, \text{different}\}$:

$$t(x, y) = \begin{cases} \text{same} & c(x) = c(y) \\ \text{different} & c(x) \neq c(y) \end{cases}$$

We can consider multiple partitions of multiple sets, where the sets may overlap. These have a correspondence with non-complete same-different graphs. It is possible to consider a same-different graph which does not represent a consistent set of multiple partitions. For example:



Logically, this is inconsistent since by transitivity of the ∼*same*∼ relation we can find,

$$\begin{aligned} &a \sim\!\textit{same}\!\sim b \sim\!\textit{same}\!\sim c \\ \Rightarrow &a \sim\!\textit{same}\!\sim c \end{aligned}$$

which contradicts

$$a \sim\!\textit{different}\!\sim c.$$

More generally, a contradiction exists when there is a different edge between two nodes connected by a path of same edges. To correct a contradiction it would be possible to either remove the *different* edge or else remove one or more *same* edges until there is no longer a path of *same* edges between the nodes attached by the *different* edge — however the choice of which facts to remove would be arbitrary. A more conservative approach is to remove facts

Table 23: An extract of data given about the predicates forming different frames in Finnish PropBank (FiPB).

lemma	frame	note	link_original	synset_id
ajaa	1	(tags: …)	none.01	02333424
ajaa	2	(tags: model:chase.01 …)	chase.01	01915724
ajatella	1	(tags: …model:ponder.01)	none.01	00599167, 02049238
alleviivata	1	(tags: …model:stress.01)	NULL	NULL
ennakoida	1	NULL	foresee.01	NULL

Table 24: An extract of data from Predicate Matrix.

11_WN_SENSE	16_PB_ROLESET
wn:chase%2:38:00	pb:chase.01
wn:chase%2:41:00	pb:chase.01
wn:stress%2:32:00	pb:stress.01
wn:stress%2:32:01	pb:stress.01
wn:foresee%2:41:00	pb:foresee.01
wn:foresee%2:31:00	pb:foresee.01
NULL	pb:ponder.01

directly participating in the contradiction: the *different* edge and every edge on every simple path between the nodes directly connected by the *different* edge. A simple path is defined as a path which does not contain the same node twice.

Here, "clustering" is used as a noun to refer to a single assignment of items to one or more clusters. Finally, we would like to go back from some consistent same-different graph to one or more clusterings. This can be done by first decomposing the graph into cliques. Each clique will end up as a single clustering. Then for each clique, all *different* edges can be removed and the resulting connected components are the clusters. Equivalently, if a node a is reachable from a node b via *same* edges, they are in the same cluster, otherwise, they are in separate clusters.

5.3.2 Automatically created gold standard

In this section, a gold standard is automatically created from existing data. The resulting gold standard only contains verbs in FiWN; It does not contain any Wiktionary definitions. The

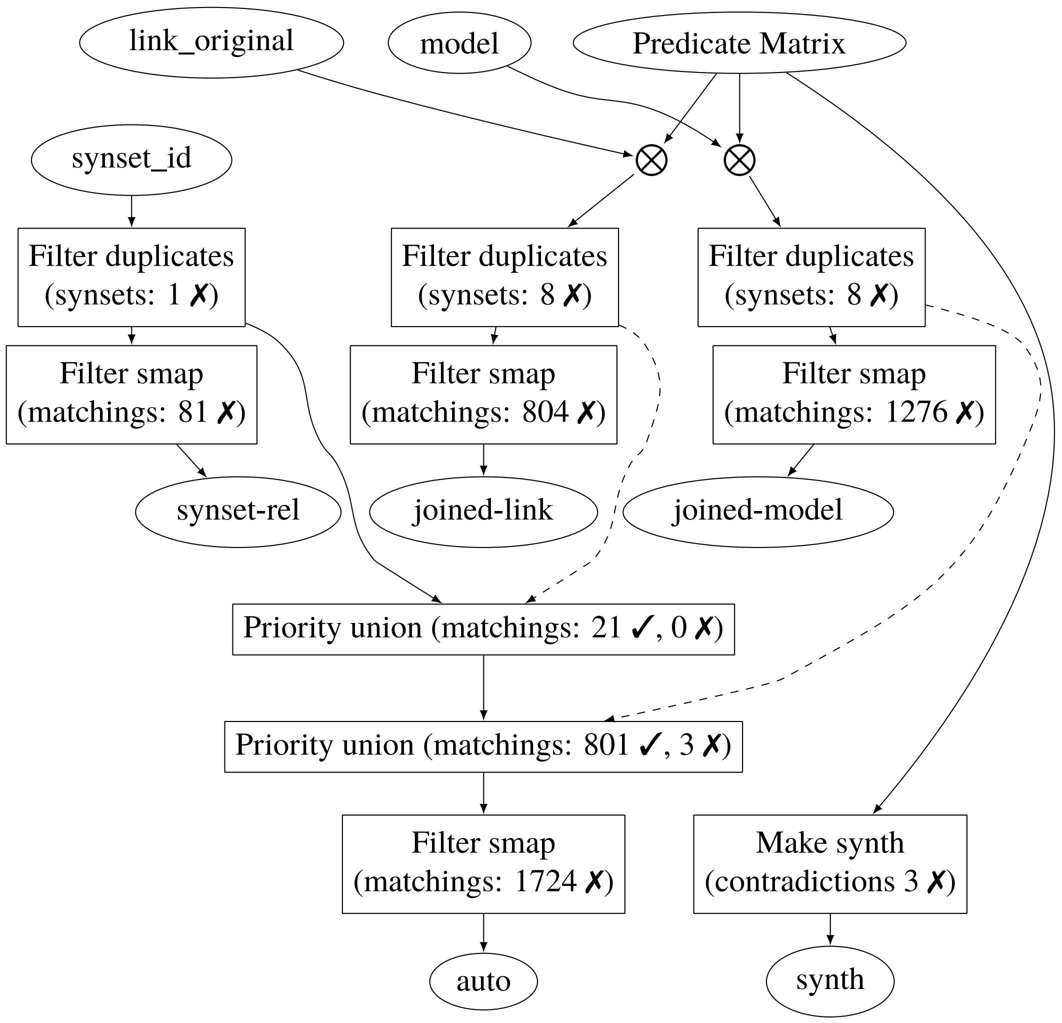

link_original
model
Predicate Matrix
synset_id
Filter duplicates (synsets: 1 ✗)
Filter duplicates (synsets: 8 ✗)
Filter duplicates (synsets: 8 ✗)
Filter smap (matchings: 81 ✗)
Filter smap (matchings: 804 ✗)
Filter smap (matchings: 1276 ✗)
synset-rel
joined-link
joined-model
Priority union (matchings: 21 ✓, 0 ✗)
Priority union (matchings: 801 ✓, 3 ✗)
Filter smap (matchings: 1724 ✗)
Make synth (contradictions 3 ✗)
auto
synth

Figure 25: Diagram showing the data flow of the creation of the automatically created gold standard. Elliptical nodes indicate some piece of data. Rectangles indicate data processing steps. The $\otimes$ symbol indicates a relational join. Dashed lines indicate the lower priority inputs to a priority union.

automatically derived gold standard is derived from FiPB and Predicate Matrix[10]. The overall data flow for creating these gold standard is summarised in Figure 25. There are three pieces of information given in FiPB which can be used to link frames to other resources, however, each piece of information is given for only some of the frames. An illustrative extract is given in Table 23. The first is that for some frames, a list of WordNet synset ids is given directly in the `synset_id` field. The second is that some frames have a `link_original` field which specifies an English PropBank frame. Finally, some contain a `model` field embedded inside their `note` field.

Three FiPB to FiWN mappings can then be derived, *synset-rel* directly from the `synset_id` field and *joined-link* and *joined-model* derived by joining respectively the `link_original` and `model` fields of FiPB with Predicate Matrix. An illustrative extract of the part of Predicate Matrix used is given in Table 24. The WordNet lemma ids given by Predicate Matrix are then mapped to synset ids using data from WordNet itself. At this point, some of the resulting clusterings may not form a partition. In particular, some synset ids may be included in multiple clusters. To correct this, for each lemma, all mentions of multiply mentioned synset ids are removed by *filter duplicates*. Figure 25 summarises how many duplicates are found in each case.

We now try to agglomerate all of the clusterings from *synset-rel*, *joined-link* and *joined-model* into a big collection of clusterings *auto*. Since all of these clusterings consist of synset ids being mapped to the same set of FiPB frames, it should be possible to simply treat each set of clusterings as a relation and take their union directly[11]. It is, however, possible for the different sources of data to disagree with one another. Therefore we make use of a *priority union* which adds rows from a secondary relation to a primary relation, as long as they do not contradict the primary relation by assigning a synset to a different frame. The priority order from highest to lowest is synset-rel, joined-link, joined-model. The reasoning for this order is: synset-rel is the most reliable since it is given directly by FiPB without needing to map through English PropBank and therefore not suffering from any semantic drift, and does not need to map through Predicate Matrix, which has less than perfect precision. Then joined-link is considered to be more reliable than joined-model by inspection. Figure 25 summarises

10. Introduced in Section 2.2.1.
11. In particular without using of any of the machinery of Section 5.3.1.

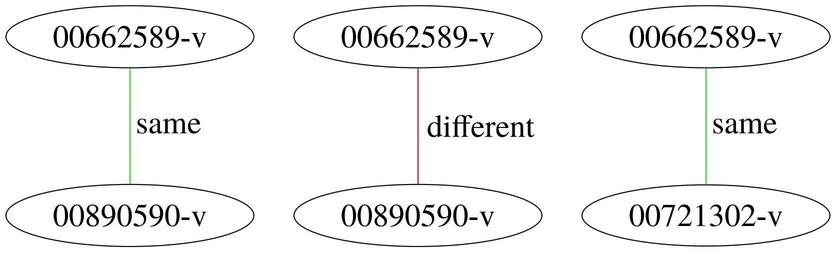
Table 25: English frame data with Finnish lemmas expanded from FiWN.

English PropBank lemma	Frame	Synset	FiWN lemmas
assure	1	00662589-v	tarkistaa …**varmistaa**
assure	1	00890590-v	luvata …**varmistaa**
insure	1	00890590-v	luvata …**varmistaa**
insure	2	00662589-v	tarkistaa …**varmistaa**
ascertain	1	00721302-v	vahvistaa …**varmistaa**
ascertain	1	00662589-v	tarkistaa …**varmistaa**

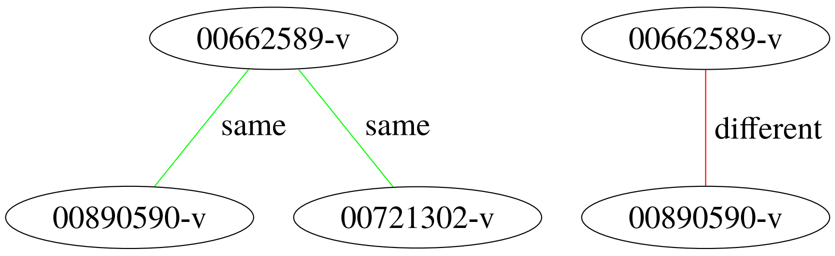
how many duplicate rows or agreements and how many contradictions were found during the priority union. Note that there were only 3 contradictions during the whole process.

The resulting joined clustering, *auto* links between FiPB lemmas, FiPB frame ids and FiWN synsets, however, it is not necessarily the case that each of the FiWN synsets contains the FiPB lemma. These matching are therefore irrelevant and so for *auto*, as well as *synset-rel*, *joined-link* and *joined-model*, all matchings which do not "self-map" are removed by *filter smap* in Figure 25.

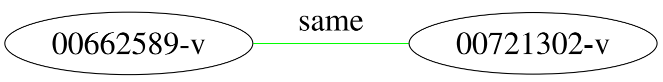
Following this line of reasoning, an extra synthetic clustering, *synth* is made directly from Predicate Matrix by considering first the clusterings resulting from grouping by English lemma, and then taking a contradiction safe union based on the same-different graph abstraction after regrouping by Finnish lemma. Consider for example getting clustering data for the lemma varmistaa. First, we look at the frame data in Predicate Matrix, and expand each WordNet synset id with a set of Finnish lemmas from FiWN as shown in Table 25 to obtain three clusterings: {{00662589-v, 00890590-v}}; {{00890590-v}, {00662589-v}}; and {{00721302-v, 00662589-v}} . We can then combine these clusterings by converting them to same-different graphs and combining them, while conservatively deleting contradictions as outlined in Section 5.3.1. First, convert each to a same-different graph:



Then take a union of all same edges and different edges separately:



Then whenever there exists a simple path between nodes directly connected by a different edge, delete the path and the different edge:



This same-different graph can be represented as a single clustering: {{00662589-v, 00721302-v}}. Only three such contradictions are found across the whole of Predicate Matrix, by *make synth* in Figure 25.

5.3.3 Manually created gold standard

In this section, a gold standard is created manually. It contains only nouns with definitions from both FiWN and Wiktionary. The *man* gold standard was created manually by the author, who is a Finnish learner at the A2 level, based primarily on the English language definitions. At a later date, the clusterings were revised with the help of a native Finnish speaker. Since the automatically created clustering focusses on verbs, the manual clustering was made up of nouns. To ensure that multiple clusters were likely, words with multiple etymology sections in Wiktionary were selected, in order of the decreasing word frequency of Speer et al. (2018)[12].

Initially, there were two annotation sessions. During the first, the most frequent 32 lemmas were annotated over about 90 minutes. In the second, at a later date, a further 96 were annotated over another 90 minutes. Later, the two annotation sets were checked, taking 45 minutes each. During this session, two clusterings were modified from the first set, and 11 were modified from the second set.

12. Obtained from `https://github.com/LuminosoInsight/wordfreq`.

A few principles were applied during annotation in order to try and create a consistent clustering:

- Keep senses from different Wiktionary etymologies in separate clusters.
  - It would be possible to violate this rule if the etymology sections were inaccurate or, for example, two distinct but related etymologies ended up producing definitions with very close meanings. Neither of these cases occurred for the manually annotated words.
- Usually, keep senses within a single Wiktionary etymology within the same cluster.
  - The usual reason that this is not done is that metaphorical extension has changed the meaning enough that it should be in a new cluster. Obvious, regular sense extensions which end up with a directly related sense such as a people/nation and their language or an activity or profession and the place where it is performed are kept within the same cluster. Non-obvious, irregular sense extensions which dramatically change the sense such as kana meaning a female chicken but also by extension meaning (in an offensive manner) an unintelligent, talkative woman are given different clusters.
- Where there exists a continuum of definitions, this can be used as justification to keep two possibly distinct senses in the same cluster. For example, the first Wiktionary etymology of *baari* has two senses: public house and cafeteria. Although these may seem quite distinct, there is a continuum of establishments from those focussed more on serving food to those focussed on serving drinks and therefore these two senses are put in the same cluster.

5.3.4 Evaluation

The gold standard clusterings used in this section, from sections 5.3.2 & 5.3.3 are summarised in Table 26.

One possible choice for an evaluation metric is to reuse precision, recall and the F_1-measure, introduced in Section 2.3.5. One way to apply these measures to clustering is to score the

name	lemmas	matchings	edges	same	diff
auto	305	839	970	699	271
synset-rel	101	259	253	107	146
joined-link	157	402	391	353	38
joined-model	149	374	345	323	22
synth	992	2568	1606	1602	4
man	128	861	4114	1731	2383
man-wn	118	460	1510	784	726
man-wiki	128	401	635	168	467
man-link	118	3938	1969	779	1190

Table 26: Summary of gold standard clusterings.

Table 27: Contingency table showing how to interpret clustering as binary classification.

		Gold	
		Same	Different
System	Same	TP	FP
	Different	FN	TN

edges of the predicted same-different graph against the gold same-different graph. Each edge in the graph is interpreted as in Table 27. Put another way, we reframe clustering as a binary classification task classifying whether each pair of examples belong in the same cluster. Snow et al. (2007) uses this method of evaluation together with the F_1-measure. Problematic with the use of the F_1-measure here is that it takes no account of True Negatives (TNs) and so becomes unbalanced: a system which puts all senses in separate clusters will always obtain an F_1-measure of 0, while a system which clusters all senses together will not. For Snow et al. (2007), the raw data set had a large number of different edges versus a small number of same edges, which mitigates the problems of using F_1-measure since a system always guessing same will obtain a relatively low F_1-measure, however as per Table 26 that is not the case here.

Also worth considering is accuracy[13], which does take account of TNs. When accuracy is applied to clustering with TPs, TNs, FPs & FNs defined this way, accuracy is equivalent to the Rand index.

13. See Section 2.3.5.

There is the possibility of adjusting the Rand index for clusterings created by chance. The adjusted Rand index ranges from -1 to 1. The expected value of the Rand index is 0 under the distribution of random clusterings. It was formulated by Vinh, Epps, and Bailey (2009) as:

$$\textit{adjusted-rand-index} = \frac{2\,(tn \cdot tp - fn \cdot fp)}{(tn + fp)\,(fp + tp) + (tn + fn)\,(fn + tp)}$$

Bootstrapping is used to obtain significance levels, and follows a similar procedure to Section 4.2.3. In this case, each bootstrapping schedule is first taken by resampling from the list of (word, cluster, word sense) tuples, hereafter referred to as cluster assignments, that make up the gold standard. For each system, each cluster assignment is scored individually by considering its edges and thus the number of TPs, FPs, FNs and TNs that it contributes to and halving it, since the responsibility for each of these outcomes is shared between two cluster assignments. The bootstrapped evaluation measure distribution is then formed by based on considering the sums of outcomes of cluster assignments chosen according to the bootstrap schedule. This procedure makes the strong assumption that each cluster assignment is independent.

5.3.5 Affinity propagation

The clustering and alignment is created using affinity propagation (Frey and Dueck 2007) as implemented in scikit-learn (Pedregosa et al. 2011). Affinity propagation takes a similarity graph and attempts to find a configuration that assigns all nodes to an exemplar such that there is a high sum of similarities between nodes and their exemplar. Exemplars are representative points which lie close to the centre of a cluster. In affinity propagation, every node starts off as a potential exemplar. The algorithm then proceeds as an iterated message passing procedure where exemplars compete to own points while receiving responsibility messages from them and points gather evidence from candidate exemplars about which is best sent to them as availability messages. As well as similarities between nodes, affinity propagation can be given different preferences for different nodes, to increase the chance they will become an exemplar.

Given many clustering algorithms have been proposed in the literature, why pick affinity propagation? The requirements here are:

R1. The clustering algorithm must work with similarity or distance matrices, rather than requiring real values vectors as inputs as is the case for k-means.

R2. The clustering algorithm should produce an exemplar for each cluster, which can be used as the definition to display to the user to represent the cluster.

R3. The clustering algorithm should not require a fixed number of clusters beforehand, but rather discover how many clusters there are naturally for each word.

Affinity propagation satisfies these requirements, however, it is not the only, nor necessarily the best, clustering method for this task. Many clustering algorithms satisfy R1, with k-means being somewhat unusual for having such a strict requirement on the format of its inputs. One example is k-medoids, which is conceptually similar to k-means and works with distance matrices. K-medoids also satisfies R2. Indeed it seems reasonable to suppose that finding an exemplar could be done as a post-processing step for any procedure which did not naturally produce them by using a graph centrality measure such as betweenness centrality on the similarity graph.

With regards to R3, a big advantage of affinity propagation is that it does not require deciding upon a certain number of clusters upfront. However, despite the fact that no particular number of clusters is chosen for affinity propagation, the number of clusters it produces is quite sensitive to the specific preference values nodes are passed, meaning we are still potentially left with the problem that we have a parameter we have to choose the value of. One possible alternative would be to run an algorithm with a fixed number of clusters such as k-medoids, for several values of k, and then choose k according to some intrinsic evaluation measure such as silhouette score. Alternatively, we could follow Galdi, Napolitano, and Tagliaferri (2014), who adapted affinity propagation to produce clusterings with a given number of clusters and then, similarly, tuned the number of clusters using silhouette score.

Instead, a small amount of tuning of a uniform preference value was performed manually. By default, the implementation of affinity propagation in scikit-learn gives each node a fixed preference set to the median similarity, but preliminary experiments revealed that this produced too many clusters, revealed as a high proportion of FNs. Thus the preference of each node is set to 0, which produced higher adjusted rand scores on the gold data. Note that strictly this constitutes parameter turning on gold data, as warned against in Section 4.2.1. However, the preference values were not turned further beyond this.

To summarise, although affinity propagation is not necessarily the best possible clustering method, it suffices for the requirements. Thus, to allow more resources to focus on comparing different ways of finding similarities between word senses, there is no comparison between different clustering methods.

Affinity propagation may not converge, to make sure a result is always available it is retried with damping levels of 0.5, 0.7 & 0.9. If all of these parameters fail to converge, all instances are placed in separate clusters.

5.3.6 Systems

Some systems use information which only available for a FiWN sense, or synset. In particular, the *Label* and *Sensevec* systems both only work in this case. Others work only for Wiktionary senses, in particular *Ety*. The other systems work for both kinds of senses, and can also be used for sense alignment.

The *Sensevec* system is based on word sense vectors. These vectors are the same Numberbatch Speer, Chin, and Havasi (2017) and AutoExtend (Rothe and Schütze 2017) vectors described in Section 4.3.4. Synset vectors are simply clustered based on cosine similarity clamped to non-negative values. Clamping was performed after preliminary experiments showed it performed on the chosen metrics.

The *Label* system uses data from OpenMultiWordNet (Bond and Paik 2012). In particular, it uses the lemma labels given to synset from many languages. Languages were filtered to only those which include at least 1000 lemmas. The reason for this is that languages WordNets smaller than this are assumed to be essentially unfinished, and so likely to just add noise. The similarity between two senses s and t in a group of senses S is then defined:

$$\text{label-sim}(s,t) = \frac{|\text{labels}(s) \cap \text{labels}(t)|}{\max_{u \in S, v \in S} |\text{labels}(u) \cap \text{labels}(v)|}$$

The *Ety* system uses etymology headings from Wiktionary and simply places all senses which occur under the same etymology heading in the same cluster (without using affinity propagation). When there is only one etymology, which is common for Wiktionary entries, all senses are placed in this cluster. Note, however, that the manually created evaluation set is not typical in this respect. It has been created only from words with a Wiktionary entry with more than one etymology section — the minority of Wiktionary entries. In addition, the etymology was used as a guide during annotation — definitions from different etymologies were generally kept separate unless there was a good reason to do otherwise.

The remaining systems work for both WordNet and Wiktionary, and can produce an alignment between them. Because of this, they all use the only common shared structure, which is the gloss of each sense.

The BERT based system, *SentBert* uses a version of BERT finetuned by Reimers and Gurevych (2019) for sentence similarity. The authors make pretrained models available. The model used was named `bert-large-nli-stsb-mean-tokens`[14]. This model starts from the larger BERT model made available by, Devlin et al. (2019) which was trained on BooksCorpus and English Wikipedia. Reimers and Gurevych (2019) then finetuned this model twice: first on a natural language inference task, and then on semantic text similarity task. Senses are clustered based on clamped cosine similarity of their glosses according to this model.

The *SoftCos* system uses the soft cosine similarity measure (Sidorov et al. 2014) as implemented in GenSim (Řehůřek and Sojka 2010; Novotný 2018). Soft cosine similarity is a generalisation of cosine similarity. Recall cosine similarity can be calculated given two column vectors u and v, with their transpose row vectors written as u' and v' as:

$$\text{cos-sim}(u,v) = \frac{u'v}{\sqrt{u'u}\sqrt{v'v}}\,.$$

14. Obtained from `https://github.com/UKPLab/sentence-transformers/blob/master/docs/pretrained-models/sts-models.md`.

Here each dimension of u and v represents the number of occurrences of a word, as in the bag of words representation. Soft cosine similarity considers u and v in as being specified in a non-orthogonal basis α with N basis vectors $e_{\alpha,1} \dots e_{\alpha,N}$ i.e., we allow that $i \neq j, e_{\alpha,i} \cdot e_{\alpha,j} \neq 0$. We can then consider the similarity between the basis vectors $s_{ij} = \cos(e_i, e_j)$. In the bag of words representation, this similarity matrix S can be constructed based on word similarity. Now to calculate $u'v$ in the basis of α we note that:

$$\begin{aligned}
u'v &= u \cdot v \\
&= (\sum_{i=1}^{N} u_{\alpha,i} e_{\alpha,i}) \cdot (\sum_{j=1}^{N} v_{\alpha,j} e_{\alpha,j}) \\
&= \sum_{i=1}^{N} \sum_{j=1}^{N} u_{\alpha,i} v_{\alpha,j} e_{\alpha,i} e_{\alpha,j} \\
&= \sum_{i=1}^{N} u_{\alpha,i} \sum_{j=1}^{N} s_{ij} v_{\alpha,j} \\
&= u'_{\alpha} S v_{\alpha}
\end{aligned}$$

And so soft cosine similarity can be defined as:

$$\text{soft-cos-sim}_S(u, v) = \frac{u'Sv}{\sqrt{u'Su}\sqrt{v'Sv}} \, .$$

Note that when S is the identity matrix, we end up with an orthogonal basis and so soft cosine similarity is equal to cosine similarity. Before calculating the soft cosine similarity, we must build a vocabulary of all words under consideration. In this case, this is repeated for each headword for which a clustering should be produced a vocabulary is built by considering the words in the glosses of all senses. S is then filled cosine similarities between these words according to the fastText MUSE vectors described in Table 12.

The *Wmd* system uses word mover's distance (Kusner et al. 2015). It builds a vocabulary for all definitions associated with each headword in the same way as *SoftCos*, but forms a cosine distance matrix, D, rather than a similarity matrix. The problem is now reduced to the Earth Mover's Distance (EMD). Intuitively, the EMD is based on the amount of work that must be done moving piles of earth from dimensions of u into holes making up each of the dimensions of v, given distances between their dimensions specified in D. To calculate EMD we must first solve the following find a flow matrix F which is a solution to the optimisation problem based minimising work, which is the Frobenius inner product of F and D:

$$\text{work}(F, D) = \sum_{i,j} F_{ij} D_{ij}$$

$$\min_F \left[\text{work}(F, D)\right] \qquad \text{such that}$$

$$F_{ij} \geq 0 \quad \sum_j F_{ij} \leq u_i \quad \sum_i F_{ij} \leq v_j \quad \sum_{i,j} F_{ij} = \min\left(\sum_i u_i, \sum_j v_j\right).$$

Here, the EMD is calculated with pyemd [15], which is a Python wrapper around the fast EMD implementation of Pele and Werman (2009). The function of this library used simply returns the work term:

$$\text{EMD}_D(u, v) = \text{work}(F, D).$$

Note that this is only a metric if the sum of the elements of u and v are the same. Normally, when preparing the definitions for use, we length normalise

$$\text{Wmd}_D(s, t) = \text{EMD}_D\left(\frac{s}{\sum_i s_i}, \frac{t}{\sum_j t_j}\right).$$

A variation system *WmdPart* performs partial matching, by computing a non-metric distance. Note the definition of EMD given here means that intuitively, once we have moved as much earth as we can, we can fill in any leftover space in the holes without any extra effort. Thus, we length normalise according to the shorter definition to allow ignoring part of the longer definition:

$$\text{min-length} = \min\left(\sum_i u_i, \sum_j v_j\right),$$

$$\text{WmdPart}_D(s, t) = \text{EMD}_D\left(\frac{s}{\text{min-length}}, \frac{t}{\text{min-length}}\right).$$

The distances are then converted to similarities using similarity $= 1 -$ distance.

15. Obtained from `https://github.com/wmayner/pyemd`.

One disadvantage of both the *SoftCos* and *Wmd* is that they do not make use of any of the synset labels from WordNet. Versions which append all English labels to the WordNet gloss are denoted *SoftCosSyn*, *WmdSyn* and *WmdPartSyn*.

Next, combination systems are created. The final system should deal with both WordNet and Wiktionary, and so one of *SentBert*, *SoftCos*, *Wmd* & *WmdPart* should be chosen. Looking ahead to the results in Table 28, we can see that for both the verb and noun datasets, *SentBert* performs best overall, while *WmdSynset* and *WmdPartSynset* are competitive for some datasets. Thus, at least one of these, denoted *Sb*, *Ws* & *Wps*, are used as the basis of all combination systems, while *WmdSynset* & *WmdPartSynset* are never combined since they are variants of the same system. These systems are incorporated with each other and *Label* by simply taking the maximum of their similarities e.g., max$(Bert, Label)$. Max was chosen following the observation that across most test sets most individual systems had a larger problem with FNs than FPs.

To incorporate the *Ety* system, we must first recast it so it can be used in our similarity graph. Here, edge weights are set to 0 between senses from different etymology sections. Preliminary experiments were conducted where edge weights were set to 1 between all senses within the same etymology section, however this decreased performance across the board. These methods are combined with others by overwriting whichever weights are set by the systems they are combined with.

Results are given in Table 28. The first horizontal line separates those systems which can only cluster one of WordNet or Wiktionary definitions from the rest. The second horizontal line separates individual systems from combination systems. Out of those systems which can cluster both WordNet and Wiktionary, cells in **bold type** have the maximum score, while those with a light blue background were not found to be significantly less than the maximum scoring system at a p=0.05 level.

We first note that within the combination systems, the *Ety+Ex* variants perform better overall than those variants without. Secondly, we note that there is no one system which performed best across all configurations or indeed best across the overall join and man datasets. We then consider those systems that perform best for one dataset. For the man dataset,

Table 28: Results of the evaluation of sense clustering systems in adjusted rand index (%). Results on the *fake* data set are shown with non-adjusted rand index (%). Systems which only work for one of WordNet or Wiktionary appear above the first horizontal rule, while combination systems appear after the second horizontal rule. Cells in **bold type** have the maximum score, while those with a light blue background were not found to score significantly less than the maximum scoring system. Significance testing is only performed for systems which support both Wiktionary and WordNet.

System	Verbs, Automatic					Nouns, Manual			
	join	ss	link	mod	**fake**	**man**	wn	wiki	link
Label	57.9	40.1	60.8	59.7	92.5	—	13.0	—	—
Sensevec	20.7	25.6	4.8	3.4	72.3	—	9.0	—	—
Ety	—	—	—	—	—	—	—	53.5	—
SentBert	47.9	51.2	23.0	13.2	81.9	39.9	45.8	17.0	41.3
SoftCos	18.0	21.1	12.4	4.3	58.3	6.9	-0.1	8.0	8.4
SoftCosSyn	31.2	39.9	11.2	11.8	71.8	16.5	11.9	9.8	19.5
WmdPart	21.8	27.5	8.1	4.4	68.1	16.0	16.4	6.0	20.1
WmdPartSyn	32.3	35.1	18.6	10.5	77.9	25.3	34.6	12.7	24.6
Wmd	27.4	34.2	15.2	4.0	69.6	9.8	2.0	9.1	15.9
WmdSyn	37.2	40.8	14.8	8.1	78.7	12.4	8.2	9.4	16.9
Sb+Ety	47.9	51.2	23.0	13.2	81.9	43.7	46.5	28.6	44.3
Sb+Ety+Ex	47.9	51.2	23.0	13.2	81.9	**51.7**	**50.4**	39.7	**53.0**
Sb+Lbl	**52.9**	**60.4**	26.4	**21.7**	**83.9**	39.2	44.5	16.7	41.6
Sb+Lbl+Ety	**52.9**	**60.4**	26.4	**21.7**	**83.9**	40.9	44.9	24.2	41.9
Sb+Lbl+Ety+Ex	**52.9**	**60.4**	26.4	**21.7**	**83.9**	46.3	44.7	35.0	48.1
Sb+Wps	44.0	43.9	28.9	15.5	83.0	28.1	40.6	10.7	26.2
Sb+Wps+Ety	44.3	43.9	28.9	15.5	82.9	44.2	47.2	36.5	42.9
Sb+Wps+Ety+Ex	44.3	43.9	28.9	15.5	83.0	50.4	46.4	48.3	52.5
Sb+Wps+Lbl	44.6	39.2	**34.0**	17.3	82.3	29.6	42.8	9.2	28.1
Sb+Wps+Lbl+Ety	44.3	39.2	33.2	16.7	82.3	41.7	45.1	35.2	40.3
Sb+Wps+Lbl+Ety+Ex	44.6	39.2	**34.0**	17.3	82.3	50.0	47.8	45.1	51.5
Sb+Ws	47.8	53.1	17.7	11.6	82.8	22.0	25.7	15.2	23.7
Sb+Ws+Ety	48.1	53.1	17.7	11.6	82.8	33.7	34.4	29.9	33.4
Sb+Ws+Ety+Ex	48.2	53.1	18.3	12.1	82.8	48.0	38.5	**59.2**	50.4
Sb+Ws+Lbl	51.6	58.7	24.5	17.0	82.7	23.9	31.6	8.1	24.6
Sb+Ws+Lbl+Ety	51.6	58.7	24.5	17.0	82.7	38.0	40.9	29.4	37.2
Sb+Ws+Lbl+Ety+Ex	51.6	58.7	24.5	17.0	82.7	42.5	36.3	45.9	44.2
Wps+Ety	32.1	34.2	18.6	10.5	77.9	42.0	40.5	39.4	43.5
Wps+Ety+Ex	32.1	34.2	18.6	10.5	77.9	45.2	40.5	45.3	48.1
Wps+Lbl	27.6	23.1	18.0	12.3	76.0	24.1	32.5	12.0	24.0
Wps+Lbl+Ety	27.6	23.1	18.0	12.3	76.1	40.7	40.1	39.4	41.0
Wps+Lbl+Ety+Ex	27.6	23.1	18.0	12.3	76.1	43.8	40.1	45.6	45.0
Ws+Ety	37.2	40.8	14.8	8.1	78.7	31.9	19.9	43.5	34.9
Ws+Ety+Ex	37.2	40.8	14.8	8.1	78.7	35.2	19.9	50.1	39.5
Ws+Lbl	44.6	35.6	30.5	17.6	77.1	8.6	9.9	8.2	8.7
Ws+Lbl+Ety	44.6	35.6	30.5	17.6	77.0	25.4	18.8	44.7	20.8
Ws+Lbl+Ety+Ex	44.6	35.6	30.5	17.6	77.0	27.9	18.8	52.8	23.0

Sb+Ety+Ex performs best scoring 51.7, and scoring 47.9 on the join dataset. For the join dataset, *Sb+Lbl+Ety+Ex* performs best scoring 52.9, and scoring 46.3 on the man dataset. We can then consider other systems on the Pareto front: those that score at least 47.9 for join and 46.3 for man of which there is one: *Sb+Ws+Ety+Ex*. So we have three potential best systems: *Sb+Ety+Ex*, *Sb+Lbl+Ety+Ex* and *Sb+Ws+Ety+Ex*.

Sb+Ws+Ety+Ex performed the best in terms of clustering between the Wiktionary part of the man dataset. A qualitative analysis of the results reveals that in the cases that *Sb+Ws+Ety+Ex* system beats the *Sb+Lbl+Ety+Ex* system for this subset of the results, most often the difference is that *Sb+Lbl+Ety+Ex* forms too few clusters. On the other hand, we have reason to prefer *Sb+Lbl+Ety+Ex* over *Sb+Ety+Ex* a priori since it incorporates the strong *Label* system, however *Sb+Lbl+Ety+Ex* performs relatively poorly on the manually created dataset.

Ultimately, it is the manually created dataset which should be given greatest consideration since it is made with the final application in mind, and for this reason, the **Sb+Ety+Ex** system is chosen the provide the final set of clusterings for usage in Chapter 6.

6 The design of TheWhatNow?!

There is increasing interest in contextualised learning of vocabulary (Godwin-Jones 2018). This chapter presents TheWhatNow?!, which is an intelligent reading assistant aiming to facilitate Finnish vocabulary learning in the context of web pages. The system presents word and idiom definitions alongside the context in which they occur. The system can be used through a web interface either as a dictionary or by manually entering or copying text into a text field, or ideally, as a browser extension to assist with reading Finnish web pages. When used as a browser extension, TheWhatNow?! presents word definitions in a sidebar. TheWhatNow?! can be classified as an ATICALL (Authentic Text Intelligent Computer Aided Language Learning) system, defined by Meurers et al. (2010) as software which produces enhanced input based on real texts.

The focus of this chapter is upon TheWhatNow?!'s simplified, unified view of the Finnish language, which nevertheless draws attention to morphological and word formation features which are useful for a language learner's development. Built upon these semantic and grammatical analyses, the user interface visualises the connection between surface forms, analytic forms and definitions.

6.1 Design criteria

In order to make a useful tool for Finnish language learners we take some knowledge or ideas comprising theories about language acquisition as a starting point, and synthesise these into design criteria used to construct the final system. This section presents and justifies the design criteria for TheWhatNow?!.

6.1.1 In context word definitions

This section presents the main function of TheWhatNow?!, which is to present word definitions in context, and justifies its design in terms of the benefit it can bring language learners. As a language learner, one of the most common things you might do when reading a text is look up new words or words that you do not remember. This process often requires a switch of context to looking up the word and then switching back. The whole process can be cumbersome, involving many steps including copying and pasting the word or retyping it as well as the context switches. Suppose we consider the speedup we can gain in the process of reading a text. We follow a similar type of reasoning to Amdahl's law (Amdahl 1967). Here we use it as a general law about speeding up a part of any multi part process rather than as a law that is only about parallelising parts of a computer program. In particular, if we speed up a fractional part p of a process by a factor of n, we gain a total speedup of:

$$\text{speedup}(p, n) = \frac{1}{(1-p) + \frac{p}{n}}$$

So for example, if looking up words takes 50% of the time spent reading a text, and it is sped up by a factor of two, then the total whole reading process is sped up by 33%. Moreover, a larger fraction of the time is now spent on the tasks of text comprehension and learning new words rather than other tasks.

6.1.2 Grammatical approach

As in Section 5.2, we aim to deal with the entirety of the rich morphology of Finnish (Karlsson 2015) and we refer to any item which can be given a definition, including lemmas, individual morphemes, combinations of morphemes and Multi-Word Expression (MWE) including schemas as *headwords*.

Why bother going to the effort to make a comprehensive treatment of word formation and complex word types? After all, these lexical items occur relatively infrequently in running text and so it may seem like a poor allocation of effort to spend time dealing with them. One assumption here is that these elements become more important after the beginner stage of language learning. If we assume a very simplified model of lexical acquisition where words are learnt in descending order of frequency, we can analyse properties of words that the language learner does not know and therefore may like to look up. Figure 26 shows two such properties varying as the number of words the learner knows increases: the proportion of all words seen which are unknown, and the proportion of unknown words which are compounds. The data is based on 1.5 billion tokens of analysed Finnish text from the Turku Internet Parsebank (Laip-

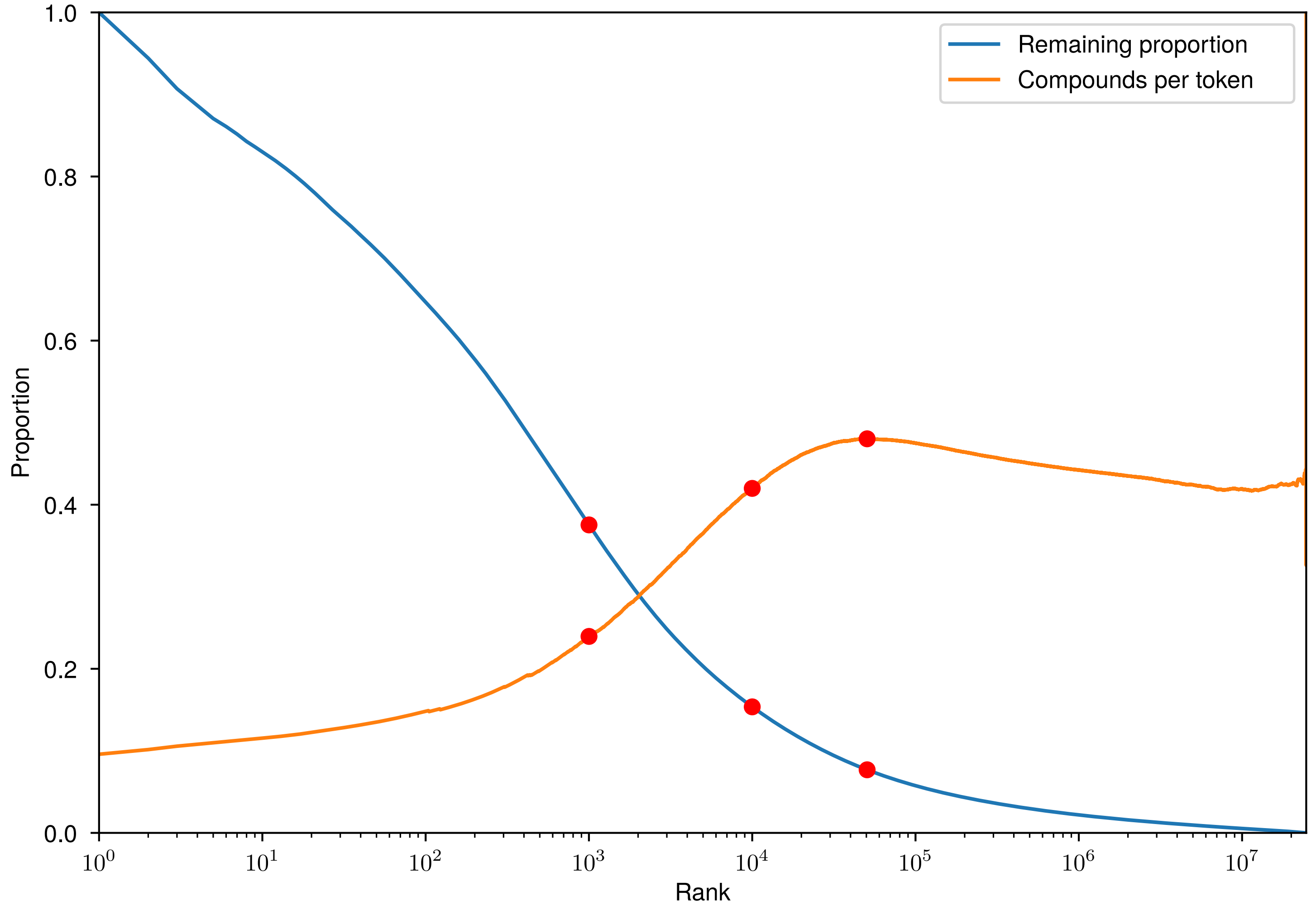


Figure 26: Proportions related to words unknown to a simplified model of a language learner. The x-axis gives the rank of the word that the learner has learned all words up to. Remaining proportion is the proportion of words in running text unknown to the language learner. Compounds per token is the proportion of unknown words which are compounds.

pala and Ginter 2014) and since there is so much text, no smoothing is performed. Taken as a whole, the corpus is 9.8% compounds. An advanced learner may know somewhere between 1 000 and 10 000 words. After 1000 words, they would be unfamiliar with 38.0% of words, which would be 23.9% compounds. After 10 000 words they would be unfamiliar with about 15.4% of words, which would be 42.0% compounds. The proportion of compounds reaches a maximum when the learner knows 50 541 words, and at this point, they would be unfamiliar with 7.7% of words, of which 48.0% would be compounds. It is assumed that other complex lexical items such as MWEs follow a similar pattern, but the analysis is not extended to include these since our ability to detect MWEs is limited by the coverage of our Lexical Knowledge Base (LKB).

Chenu and Jisa (2009) and MacWhinney (2008) note that while in the early stages of learning a language, it may be useful to employ links between concepts from the mother tongue onto the target language, at later stages, it becomes important for learners to begin to form links between concepts in the target language without direct reference to the mother tongue. As noted, Finnish contains many word associations that are marked as part of word formation, readily available to the language learner.

Lightbown and Spada (2013, pp. 168–175) state that input alone is not sufficient for language acquisition in older learners. Key to this is the idea that attention is required for new linguistic knowledge to enter the memory, formulated in the noticing hypothesis by Schmidt (1990). Following this concept, systems such as those of Meurers et al. (2010) and Reynolds, Schaf, and Meurers (2014) were created to automatically enhance input in web pages in order to promote noticing. These systems draw attention to, for example, the part of speech of a word. TheWhatNow?! draws attention to the connection and overlap between analytic and surface forms, to promote learning of morphology, as well as to the formation of the word itself, to help the formation of morpho-semantic links in the target language.

Finnish is quite agglutinative, meaning it is reasonably normal for morphemes to preserve most of their surface form when combined. However, sound changes, as well as the sheer number of morphemes that can be included in a word (as marked delimited by spaces in text), can provide a barrier for learners to begin to find familiar lemmas and morphemes. Any kind of segmentation and normalisation could help learners find familiar lexical items.

A lot of reference material for Finnish makes use of rather technical language by, for example, using latinate names for case endings. Bleyhl (2009) notes that treatments of language which are heavy on grammatical analysis and the associated linguistic terminology can be counter productive in language instruction since they waste time on material other than the comprehensible input needed for true language acquisition. This large amount of extra material can lead to reduced confidence from learners.

The approach taken here is to treat as many morphemes as possible as lexical items. Taking case endings as an example and assuming English as a mother tongue language[1], most Finnish case endings have a fairly good correspondence in terms of function with prepositions in English. Since many of the case endings, most notably the locative case endings, occur mostly in the same form at all times, morphemes are referred to by a normalised form. So for example rather than saying ‘voileipäkakusta‘ is in elative form, we just present the fact that -sta is one of its constituent morphemes. The names of the most common case endings, partitive and genitive, are named in the interface on the basis that their usage is more grammatical. That is to say that it is more often the case that their usage is obligated rather than they are used intentionally to convey extra information. Plural is also named on the basis it is likely to be familiar from English.

No major distinction is made between different types of morphemes in the user interface, to present the rich morphology of Finnish in a way which allows a learner to find familiar morphemes, promotes attention to word formation, and doesn’t require the introduction of too much terminology. All morphological analyses are normalised segmentations.

6.2 Analytical segmentation of Finnish

This section develops *asafi*: An **a**nalytical **s**egmenter with **a**lignments for **Fi**nnish. Asafi is made with the particular requirements of TheWhatNow?! in mind. The main requirements are:

1. The assumptions will hold for many other languages also.

R1. Produces analytical or normalised forms of grammatical morphemes. This means, for example, segmenting kakusta as kakku -sta rather than kaku -sta. As discussed in Section 6.1.2, this allows a less technical presentation of word structure.

R2. Produces multiple ranked analyses so as to degrade gracefully when for example an MWE is spuriously found or a Part Of Speech (POS) tagger produces an incorrect result. This means that we would like for example kuusi to be segmented into both kuusi (spruce tree) and kuu -si (moon + 2nd person possessive; your moon).

R3. Produces analyses which are as segmented as possible. This means that we should segment a word like voimakas as far as voida -ma -kas. This shows the learner the internal structure of the Finnish lexicon as much as possible.

R4. Produces analyses in which the final segmentations are related with all lexical items that make them up, so even though voileipäkakusta is segmented as voi leipä kakku -sta, we also want to know that voileipä and voileipäkakku are constituent lemmas. This means we can display all relevant word definitions.

R5. Where possible, attempts to add tags to the grammatical morphemes, such as POS tags for lemmas, since this will allow associating only relevant definitions with the morpheme.

R6. Produces an alignment between the surface and analytical forms. Since this information will be used to draw attention to form, as discussed in Section 6.1.2 in the user interface shown in Section 6.3.

Because of R1, approaches such as Morfessor (Creutz and Lagus 2005) (Virpioja et al. 2013) or Omorfi's segmenter which produce grammatical surface morphemes are not appropriate. Approaches such as Byte Pair Encoding (BPE) (Sennrich, Haddow, and Birch 2016) produce segmentations which may not consist of grammatical morphemes and are thus also not useful in this case. It may seem that we must simply map from Omorfi's analyses to analytical segments, however, Omorfi's analyses do not deal with most derivational morphemes, instead, it adds derived word forms into its full form list of lemmas, meaning it does not meet R3.

This task is quite similar to that of Kann, Cotterell, and Schütze (2016), who use an encoder-decoder model followed by a reranker to produce normalised segmentations of English, German and Indonesian. However, the task here is broader due to the additional requirements of R4, R5 and R6. Additionally, Finnish tends towards a greater number of morphemes per token than these languages, having a similar level of compounding to German, but a greater number of inflectional and derivational morphemes.

The overall approach is somewhat related to that of Shapiro (2016), who starts from segmentations produced by Morfessor 2.0 (Virpioja et al. 2013), and classifies whether boundaries between segments are compound boundaries or not. Note that constituent words of compounds are a case where analytical segments are the same as surface segments since compounding does not usually cause sound changes in Finnish. There, the task is solved using a mix of language modelling and morphophonological constraints.

Non-goals are:

- Morphological generation. There is no need to go from analyses back to surface forms for our purposes.
- Optimised implementation. There is no need for the implementation to be fast enough to analyse large batches of text quickly, but only small groups of words at a time. Therefore the system may be built by using a slow language like Python to integrate the results of other systems.

The basic approach is to combine analyses from Omorfi's analyser and information from Wiktionary together to produce segmentation trees. Grammatical constraints are used to remove spurious analyses. The analyses are coupled with alignment information. In some cases, extra hard coded rules are used to produce the desired type of analytical segmentations.

6.2.1 Normalised segments from Omorfi and Wiktionary

Omorfi's morphological analyser produces analyses in its own format, which has some degree of compatibility with tags from the universal dependencies project (Pyysalo et al. 2015). As an example, kakusta may be analysed as `[WORD_ID=kakku][UPOS=NOUN][NUM=SG][CASE=ELA]`. A series of mappings for each category map relevant tags to analytical morphemes so that `[CASE=ELA]` is output as -sta, while `WORD_ID` is passed through. The order in which the tags appear is the same order as the surface morphemes appear, meaning our analytical morphemes are in the same order as the surface morphemes. This mapping is part of finntk.

Wiktionary contains various tags which give information about word formation. The Wiktionary scraper of Section 5.1 adds this information to its database so that each etymology section can have a piece of derivation information. While there are quite a few different template tags used for etymology, they are normalised so that every etymology is either an inflection, derivation or compound consisting of normalised segments. For compounds, the normalised segments are given as arguments to the template tag, and this is also the case for most derivation template tags. There are some derivation template tags which must be mapped to normalised segments, such as the `agent noun of` template tag, which is mapped to "-ja". Finally, the `form of` template tag makes use of grammatical terms such as elative, and these are mapped to normalised segments such as "-sta". This is done as part of normal wikiparse processing, but the mapping data is part of finntk.

6.2.2 Building a segmentation derivation tree

At first glance it would appear that as far as obtaining normalised segments is concerned, the process is already finished at this point. All we need to do is look up our word with either the from either the Wiktionary or Omorfi data and we are done. However, complex words may have several levels of compounding or word derivation and inflection. Thus, we may have to make use of several lookups to fully segment a word form. We also want to make sure a completely segmented word form can be associated with all lexical items that make it up. We thus shift our perspective to think of these analyses as rules and the segmenter as a rule engine which applies them to produce derivation trees subject to constraints. Each rule can match any segment and produce many segments.

The basic rule engine operates by recursively applying rules. It keeps track of the current front of the derivation tree. At each iteration, each node from the front is considered and one or more steps consisting of applying one or more rules are taken to create child nodes from each node to create a new front. There may be multiple rules which can match a segment. In this case, all combinations of rules matching each matchable segment are applied. When either there are no more rules which match, or there is a match which does not expand any segments, the node is marked as terminal. Note that a rule may match, but not produce any more segments, instead explicitly preventing further segmentation. Consider for example leipä, which has a Wiktionary entry with two etymologies: leipä as a noun (bread), and leipä as a Hawaiian lei + the enclitic particle -pa/-pä. The first etymology produces a rule which matches leipä but is marked as not performing any further segmentation.

A simple approach would be to allow all rules to apply at once. However, Omorfi analyses do not work very well as rules as is in our case, for example for voileipäkakusta Omorfi produces the following analyses:

```
[WORD_ID=voi][UPOS=NOUN][NUM=SG][CASE=NOM] ⌋
 ↪  [BOUNDARY=COMPOUND][WORD_ID=leipä][UPOS=NOUN][NUM=SG] ⌋
 ↪  [CASE=NOM][BOUNDARY=COMPOUND][WORD_ID=kakku][UPOS=NOUN] ⌋
 ↪  [NUM=SG][CASE=ELA]
[WORD_ID=voileipä][UPOS=NOUN][NUM=SG][CASE=NOM] ⌋
 ↪  [BOUNDARY=COMPOUND][WORD_ID=kakku][UPOS=NOUN][NUM=SG] ⌋
 ↪  [CASE=ELA]
[WORD_ID=voileipäkakku][UPOS=NOUN][NUM=SG][CASE=ELA]
```

If we were to apply each of these analyses as rules we would end up with 3 final segmentations. However, for our purposes, they are all the same analysis, but with lexical items made from multiple segments from the final analysis attached, as per R4.

Therefore we take the following approach:

- First, apply Wiktionary based rules recursively.
- Fetch all Omorfi rules resulting from looking up the whole word form (i.e., no recursive lookup is performed)
- While there are Omorfi rules left:
  - Try to match all Omorfi rules with an existing node using the rule's output. Those which matched are removed from consideration.
  - If any rules did not match, apply the shortest ones starting from the root node and discard.
  - Apply Wiktionary based rule recursively.

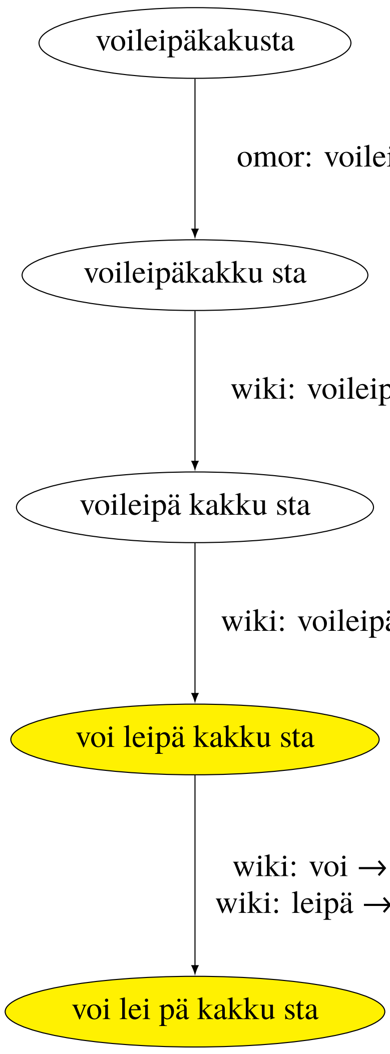


Figure 27: Derivation tree produced by asafi for voileipäkakusta. Terminal states, corresponding to outputs are highlighted in yellow. In this example, leipä is incorrectly split into lei and pä.

- Apply any reverse rules (see Section 6.2.4)

An example of a tree produced using this approach for voileipäkakusta is shown in Figure 27.

6.2.3 Constraints upon rules

Applied as is, this scheme will produce a large number of impossible segmentations. This is because each segment is considered only as a string. However, if we consider the POS of each segment, we can place constraints to avoid some impossible segmentations. Consider Figure 27. At the second from last node, voi could be considered as a form of ”voida”.

Indeed there is an etymology entry from Wiktionary for this case. However, we can consider the space of possible compound word POS patterns to narrow this down.

We start by moving everything to a common set of POS tags. The tags are based on the POSs in Princeton WordNet (PWN) and are: Verb, Noun, Adverb, Adjective & Unknown. Universal dependency POS tags are mapped into this schema so that Omorfi analyses can be used. Wiktionary POS headings are also mapped. Since this set is so small, multiple source POS tags are mapped onto a single target POS. For example, adpositions (prepositions or postpositions) are mapped onto the Adverb class. All closed classes, as well as interjections and the prefix heading from Wiktionary, are mapped to Unknown. Note that constituent words of Finnish compounds can be inflected words, and so here inflected forms are treated as having the POS of their lemma.

The permissible compound POS patterns can then be produced by a list of production rules, obtained by studying Hyvärinen (2019):

Verb → *Noun Verb* (e.g., koe+lentää)
Verb → *Adverb Verb* (e.g., edes+auttaa)
Noun → *Noun Noun* (e.g., voi+leipä)
Noun → *Adjective Noun* (e.g., puna+viini)
Adjective → *Adjective Adjective* (e.g., hyvän+näköinen)

Note that verb and adjective compounds are much rarer than noun compounds in Finnish. We usually start by treating the whole token as having Unknown POS. Unknown acts as a wildcard, able to match any POS. We may consider a segment as having one of a set of POS tags. In case we receive information splitting a compound into more than two parts, we can obtain the possible POS patterns by producing a set of expanded rules by applying the grammar formed by the production rules.

We now follow a worked example following the derivation path of Figure 27:

1. We start with voileipäkakusta, which has Unknown POS.

2. We receive an Omorfi rule which removes the inflection to produce voileipäkakku -sta. This is permitted since Unknown can match anything. We label voileipäkakku as a Noun based on Omorfi's tag.
3. We receive a segmentation of voileipäkakku (Noun) as voileipä kakku from a Wiktionary rule. This is permitted because we know voileipäkakku is a Noun. Based on the production rules we know that voileipä may be an Adjective or a Noun and kakku must be a Noun.
4. We receive a segmentation of voileipä (Noun) as voi leipä. This is permitted since voileipä is a Noun or an Adjective. Based on the production rules we know that voi may be an Adjective or a Noun and leipä must be a Noun.
5. We receive an analysis of voi (Verb) as voida (3rd pers.) from a Wiktionary rule. This is not permitted since voi may only be an Adjective or a Noun.
6. We receive two analyses of leipä (Noun):

   (a) We receive one etymology from Wiktionary which says that leipä (Noun) can exist with no further segmentation and so the current node is set as a terminal. This is permitted since the POS matches.

   (b) We receive another etymology which has a segmentation of leipä (Noun) as lei pä. This is permitted since the POS matches — although the resulting segmentation of voileipäkakusta is spurious. Since there are no more rules to match, the new node is set as a terminal.

6.2.4 Producing alignments

As per R6, we would like to produce alignments between surface forms, headwords and analytical morphemes to be used in the user interface shown in Section 6.3. The alignments consist of two spans: a surface span, denoted with strong highlighting in the user interface, and a logical span, denoted with weaker highlighting in the user interface. An analytical morpheme produces a logical span across that span of the surface form or headword which is logically responsible for it. In linguistic terms, it should cover exactly its corresponding morpheme or allomorph within the surface form. In all cases, every surface character should be covered by at least one logical span. In most cases, logical spans should be disjoint. Indeed,

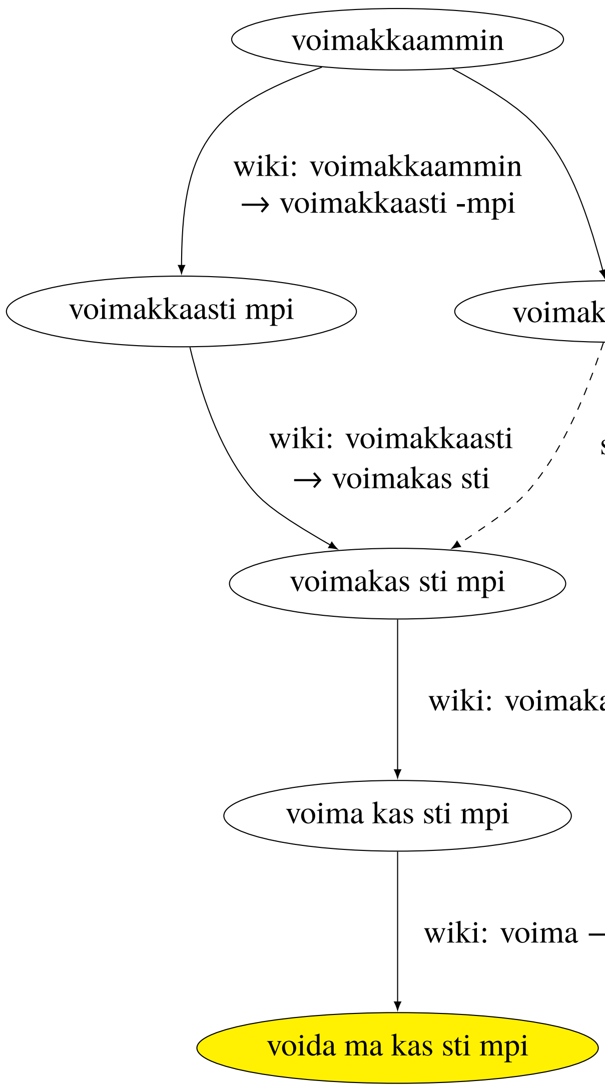


Figure 28: Derivation tree produced by asafi for voimakkaammin. Terminal states, corresponding to outputs are highlighted in yellow. The dashed line denotes the retroactively added synthetic node is a secondary parent of its child, thus allowing the structure to be used as a tree or a directed acyclic graph.

this is a feature of agglutinative languages: a one to one correspondence between the grammatical features of a word form such as case and number and morphemes. However, some morphemes in Finnish are more fusional such as -mme which encodes person and number within one morpheme and -mmin which is logically the combination of the adverb forming derivational morpheme -sti and the comparative -mpi. These are handled on a case by case basis: with -mme left as is due to how common it is, and -mmin broken into -sti and -mpi as a special case, with both logical spans covering the whole fusional surface morpheme. In the case of -mmin, we retroactively add a second tree producing a directed acyclic graph structure see Figure 28.

In the common case, we produce a logical alignment by working backwards from each final analytical morpheme through the derivation path leading back to the surface form. At each step of the derivation path, we first consider each of the rules applied to reach that step from its parent. If a segment is unchanged by a rule, there is no change to its logical span. If there is a rule, and it is not a rule which gives any alignment information, the Force-Align procedure is used to attempt to find the logical spans of each child segment given the parent segment.

Force-Align aims to find an alignment consisting of a matching between child segments and spans across parts of the parent segment, such that a prefix of each child segment is matched with ordered, non-overlapping spans of the parent segment — with gaps between the spans of the parent being possible. Matches are performed after normalisation. All strings are lowercased and the front vowels ä, ö and y are mapped to the respective back vowel a, o and u. As defined so far, Force-Align could trivially just return a list of empty spans, however, we would additionally like to skip as few characters as possible, and so we define a cost for solutions so as to prefer lower cost solutions over higher cost ones:

$$\text{cost} = (\text{parent characters skipped})^2 + (\text{child characters skipped})^2$$

An example showing the type of alignments produced by Force-Align made between each generation of the derivation tree of voimakkaammin (previously shown in Figure 28) is shown in Figure 29.

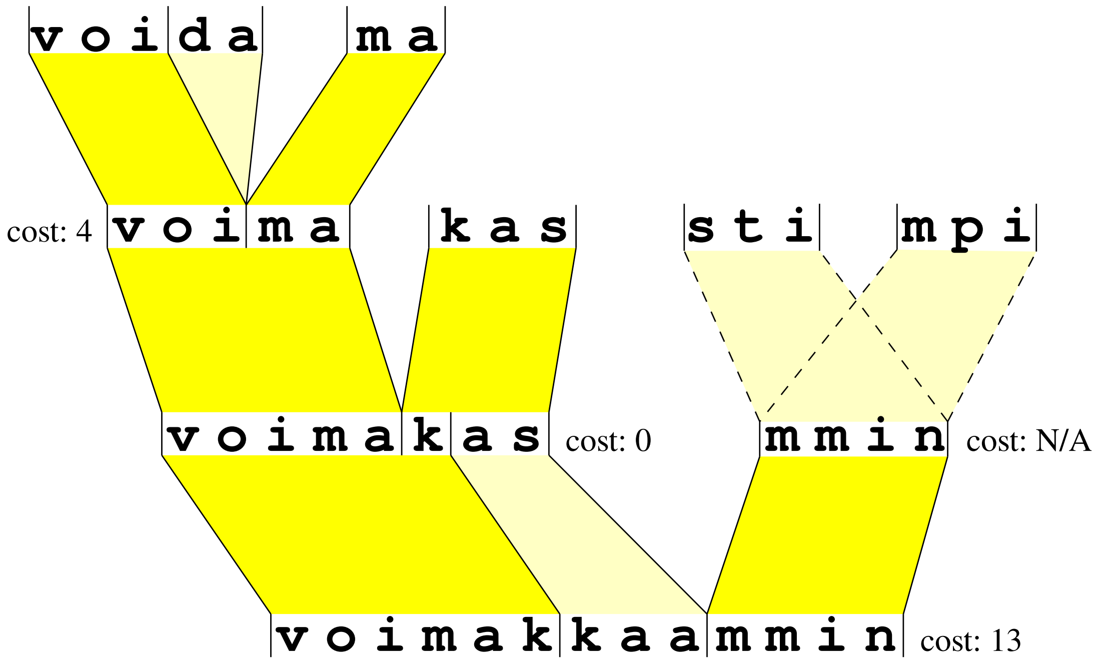


Figure 29: Alignment of voimakkaammin across multiple generations of the derivation tree of Figure 28. The tree is drawn upwards instead of downwards to emphasise the path from child back to parent. Dark yellow portions denote surface spans, while each of the whole yellow portions including dark and light denote the whole logical span. The cost of each alignment according to Force-Align is shown next to the parent segment. The dashed lines indicate the alignment is not produced by Force-Align but instead obtained from the underlying rule. In this case: the synthetic rule -mmin → -sti -mpi.

```
Function FORCE-ALIGN
 (normalised parent string $p$, array of normalised children strings $c_1 \ldots c_n$ )
 returns alignment $a$
    Create a bounded priority queue $pq$ with the lowest cost paritial solution at its front
    Add an empty partial solution into $pq$
    while The solution at the front of $pq$ is not complete do
        Take a partial solution $j$ from the front of $pq$
        /* Try to make a match                                          */
        if $j$'s cursor into $c$ has not reached end and $j$'s cursor into $p$ has not reached end
          and $p_{(j\text{'s cursor into } p)} = c_{(j\text{'s cursor into } c)}$ then
            Add copy of $j$ into $pq$ with its cursors into $p$ and $c$ incremented
        end
        /* Try to skip a parent character                               */
        if $j$'s cursor into $p$ is not at beginning or end then
            Add copy of $j$ into $pq$ with its cursor into $p$ and its *parent characters skipped*
              incremented
        end
        /* Try to skip the rest of the current child segment
           */
        if $j$'s cursor into $p$ is not at beginning and $j$'s cursor into $c$ has not reached end
          then
            Add copy of $j$ into $pq$ with its cursor into $c$ and its *child characters skipped*
              incremented
        end
    end
    $a$ := alignment formed by solution at front of $pq$
end
```
Algorithm 4: The Force-Align procedure to find an alignment between a parent string and its segmented children strings.

Force-Align is implemented as a dynamic programming style procedure given as pseudocode in Algorithm 4. At each step, Force-Align keeps track of candidate solutions in a priority queue, with the lowest cost partial solution always being at the front. The priority queue is bounded at length 100 to bound the running time — making the procedure a form of beam search. Whenever a partial candidate solution is taken from the front of the queue, all applicable out of three possible new partial solutions are created and added to the priority queue: making a single character match; skipping a single character from the parent string; and skipping the rest of the characters in the current child segment. The procedure ends when there is a complete solution at the front of the queue. Each child segment's logical span covers the characters from the beginning of its first character match to just before the first character match of the next child segment, or until the end of the parent segment in the case of the last child segment.

Next, we extend the logical alignment that now exists between segments separated by a single generation in the derivation tree to multiple generations. We do this by at each generation considering a current working logical span together with the logical span formed by the current child segment onto the parent segment. First, the working span is shifted right according to the position of the left edge of the logical span of the parent segment. Next, we consider whether the original segment is still on the right edge of the current child segment. This is the case to begin with and continues to be the case as long as we have followed the rightmost child segment previously. If it is on the right edge, then we replace its right edge with the right edge of the parent segment. If not, we leave its right edge as is.

Consider again the example of Figure 29 and the analytical morpheme -kas. It begins on the right edge. We consider its alignment onto voimakas and find we must shift its left edge by five characters. We replace its right edge with a that of its parent which happens to be identical to its current right edge. -kas is still on the right edge of voimakas when we consider the alignment of voimakas and voimakkaammin. At this step, we do not have to shift the left edge. The right edge is replaced with the right edge of the alignment of voimakas onto voimakkaammin, shifting it right by one character. The final span contains the characters 'kkaa' — which is the allomorph corresponding to 'kas', as required.

```
kas
 kas
  kas
   kas
kkaa
```

Figure 30: Possible surface alignments of ‘kas’ onto ‘kkaa’. Aligned characters are indicated in red.

We now turn to the problem of finding surface spans. This is done by moving the child segment across the part of its parent segment which it logically spans and choosing the position with the longest common prefix. For example, we choose the second alignment shown in Figure 30. Notice that surface alignments already exist in Figure 29 which could be readily used, however performing a second surface matching pass produces superior results. The first pass, performed as a side effect of Force-Align, chose the first option of Figure 30, but in cases such as these, where a strong grade kk is produced from a weak grade k we should prefer solutions similar to the second option. An additional benefit is that this scheme extends nicely to situations where Force-Align is not needed, such as when logical span information is available from Wiktionary – although this information is not used by the current version of asafi.

6.2.5 Aligning MWEs and schemas

We can now produce normalised segmentations of individual tokens from some text and obtain alignments of the analytical morphemes with the surface token. We can also obtain alignments of the analytical morphemes with all headwords in its derivation path, as per R4. However, for the purposes of TheWhatNow?! we need to extend these abilities to deal with the MWEs obtained in Section 5.2. In particular, we should like to be able to treat the extracted MWEs as headwords and thus obtain alignments between them and analytical morphemes.

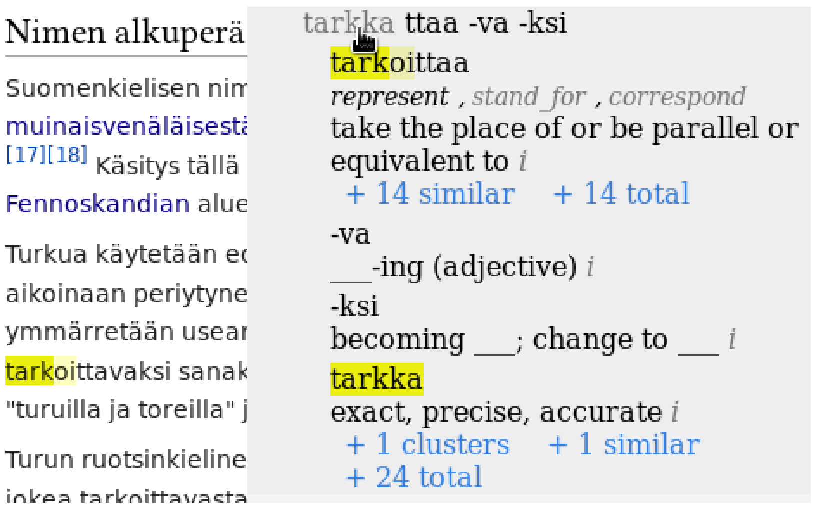
Aligning analytical morphemes with MWE headwords is done by starting from derivation trees of each surface token and aligning each token within the MWE headword with them as a second step. Each token within the MWE is accompanied by an Omorfi analysis, and thus can be segmented by mapping it through the same map mentioned in Section 6.2.1. In the case of wildcards, we are always dealing with the wildcard plus a single morpheme and so segmentation is trivial. For each MWE token we can then find steps in the derivation tree in its corresponding surface token it is compatible with. We walk down the derivation tree, and if any node has an ordered superset of the segments of the MWE token then the whole subtree is compatible. A mapping between the segments in the MWE token and the node within the derivation tree can then be created. Now highlighting parts of an MWE headword based on an analytical morpheme is simply a matter of going ascending the derivation tree. If we hit a node associated with an MWE then we can map from the analytical morpheme to the corresponding MWE morpheme.

In the user interface, it is the terminal segmentations which serve as section headings for headword definitions. However, it would not make sense in terms of hierarchy to display the MWE headwords under only one of the tokens which make it up. Thus we would like to be able to create segmentations which span multiple tokens. We do this by starting from the MWE and taking all combinations of terminal segmentations of each non-wildcard token. Any wildcard tokens are not expanded.

6.3 User interface

A screenshot of the user interface is shown in Figure 31. Definitions are grouped by normalised segmentation. Within each normalised segmentation there are defined headwords, each corresponding to one or more of the normalised segments. They are ordered in decreasing order of coverage of the normalised segmentation, meaning those definitions which define the meaning of the surface form most closely appear closest to the top. Within each defined headword appears one or more clusters of definitions, each with an exemplar.

To bring attention back to surface forms from the normalised forms, the interface highlights the surface forms, as the learner hovers over the segmented forms. See Figure 32. The interaction recalls a one dimensional “hover scrub” action. Initially, the whole word or phrase is lightly highlighted. As the learner scrubs over analytic morphemes, the surface span of



Nimen alkuperä

Suomenkielisen nim
muinaisvenäläisestä
[17][18] Käsitys tällä
Fennoskandian alue

Turkua käytetään ec
aikoinaan periytyne
ymmärretään usear
tarkoittavaksi sanak
"turuilla ja toreilla" j

Turun ruotsinkieline
jokea tarkoittavasta

tarkka ttaa -va -ksi

tarkoittaa
represent , *stand_for* , *correspond*
take the place of or be parallel or equivalent to *i*
+ 14 similar + 14 total

-va
___-ing (adjective) *i*

-ksi
becoming ___; change to ___ *i*

tarkka
exact, precise, accurate *i*
+ 1 clusters + 1 similar
+ 24 total

Figure 31: A screenshot showing the Finnish Wikipedia page *Turku* being read using the browser extension.

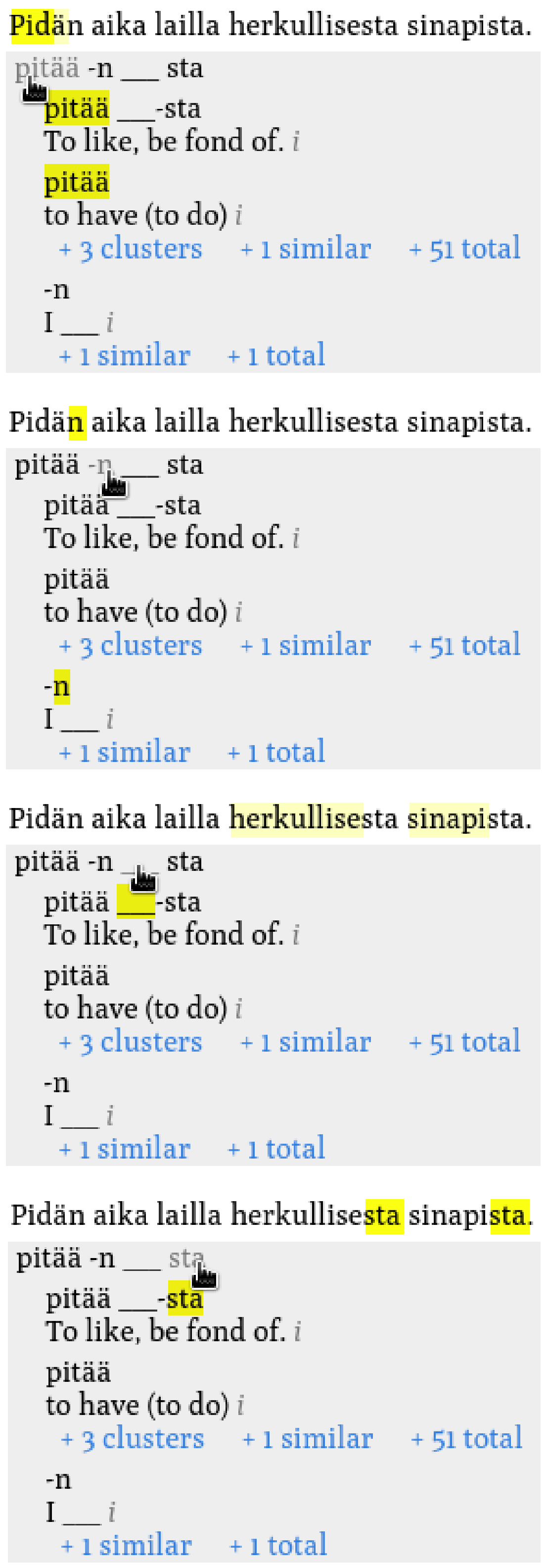
the surface form is highlighted in a dark shade, while the remainder of the logical span is highlighted lightly.

To show the connection between the normalised segmentation and its definitions, parts of the defined headwords are highlighted when normalised segments are hovered over, as shown in Figures 31 & 32. The whole interaction serves to link the different views of surface form, analytical form and headwords.

6.4 Architecture

This section gives an overview of the architecture of the final TheWhatNow?! web application and browser extension. A block diagram of the main architectural components is shown in Figure 33. The web application is built on top of Nuxt[2]. Nuxt has both client and server components allowing a Vue[3] based front end to be pre-rendered before delivery to the browser. Both of the web application and browser extension clients connect to the broker over a WebSocket, which acts as the main entry point to the server side. The WebSocket abstraction provides reliable, message based two way transmission of text or binary data. Here, type tagged JavaScript Object Notation (JSON) messages are sent. The WebSocket connection is

2. See `https://nuxtjs.org/`.
3. See `https://vuejs.org/`.


Pidän aika lailla herkullisesta sinapista.

pitää -n ___ sta

pitää ___-sta
To like, be fond of. *i*

pitää
to have (to do) *i*
+ 3 clusters + 1 similar + 51 total

-n
I ___ *i*
+ 1 similar + 1 total

Pidän aika lailla herkullisesta sinapista.

pitää -n ___ sta

pitää ___-sta
To like, be fond of. *i*

pitää
to have (to do) *i*
+ 3 clusters + 1 similar + 51 total

-n
I ___ *i*
+ 1 similar + 1 total

Pidän aika lailla herkullisesta sinapista.

pitää -n ___ sta

pitää ___-sta
To like, be fond of. *i*

pitää
to have (to do) *i*
+ 3 clusters + 1 similar + 51 total

-n
I ___ *i*
+ 1 similar + 1 total

Pidän aika lailla herkullisesta sinapista.

pitää -n ___ sta

pitää ___-sta
To like, be fond of. *i*

pitää
to have (to do) *i*
+ 3 clusters + 1 similar + 51 total

-n
I ___ *i*
+ 1 similar + 1 total

Figure 32: A composite screenshot showing different stages of the interaction resulting when a user brushes over segments in the text analyser.

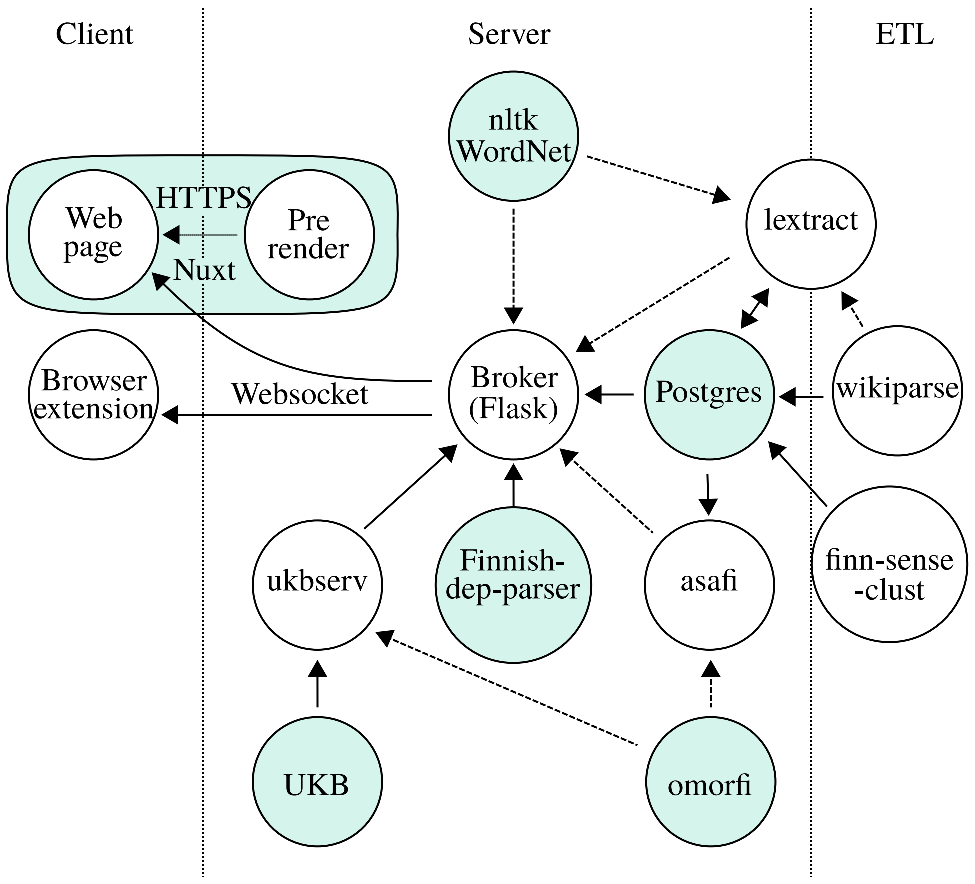


Figure 33: The architecture of TheWhatNow?! as a block entity diagram. The arrows show the direction of travel of linguistic analyses. Solid arrows designate communication that happens across processes, while dashed arrows happen between a program and a library, within the same process. The diagram is separated into three sections. Clients run on the user's device. Servers run on a server machine concurrently with the clients, providing linguistic information to the clients. Extract Transform Load (ETL) batch jobs also run on the server machine as one off or occasional tasks, independently of clients. Components with a light green background are external and were not written as part of this thesis.

stateful; The broker keeps track of state related to the document the user is currently using the client with.

When a user requests an analysis for a word in a document, the first thing the client does is push the paragraph or similar fragment in which the word is contained to the broker in a `ParagraphPush` message. As part of this, the client computes a hash of the paragraph to use as an identifier in later requests. This is immediately followed by a `CursorAnalysis-Request` message specifying the paragraph and character position to be analysed. If the user requests another word analysis from the same paragraph, only a `CursorAnalysis-Request` message will be sent, since the client knows the broker has already been sent the paragraph and thus already has access to the paragraph and possibly some linguistic analyses related to it. When the paragraph will no longer be used, for example, the language learner navigates away from the page, a `ParagraphInvalidate` message is sent to tell the broker it can delete the paragraph and any analyses of it it has performed. The client can also send a `SingleAnalysisRequest` message containing a single word. This is used by the single word lookup page on the web page client.

In response to a `CursorAnalysisRequest` or `SingleAnalysisRequest`, a stream of replies wrapped in JSON container tagged as either `CursorAnalysisResponse` or `SingleAnalysisResponse` are sent containing: word definitions from Wiktionary[4] or FinnWordNet; sense clusterings[5]; sense weights from UKB[6]; any MWEs at the user's cursor[7]; and aligned analytical segmentations from asafi[8]. The client then displays this information to the language learner. The overall flow of this linguistic information on the server side is shown in Figure 34.

The broker is free to eagerly perform tasks for a whole paragraph at a time, or else to delay them until a `CursorAnalysisRequest` comes and potentially only perform the part of the task relevant to the cursor position, or to mix both strategies. Currently, tokenisation, dependency parsing, MWE extraction and Word Sense Disambiguation (WSD) are per-

4. Scraped with wikiparse in Section 5.1.
5. Created with the Sb+Ety+Ex system from Section 5.3.
6. Determined to be the best of the evaluated systems overall in Chapter 4.
7. Extracted using lextract; See Section 5.2.
8. Developed in Section 6.2.

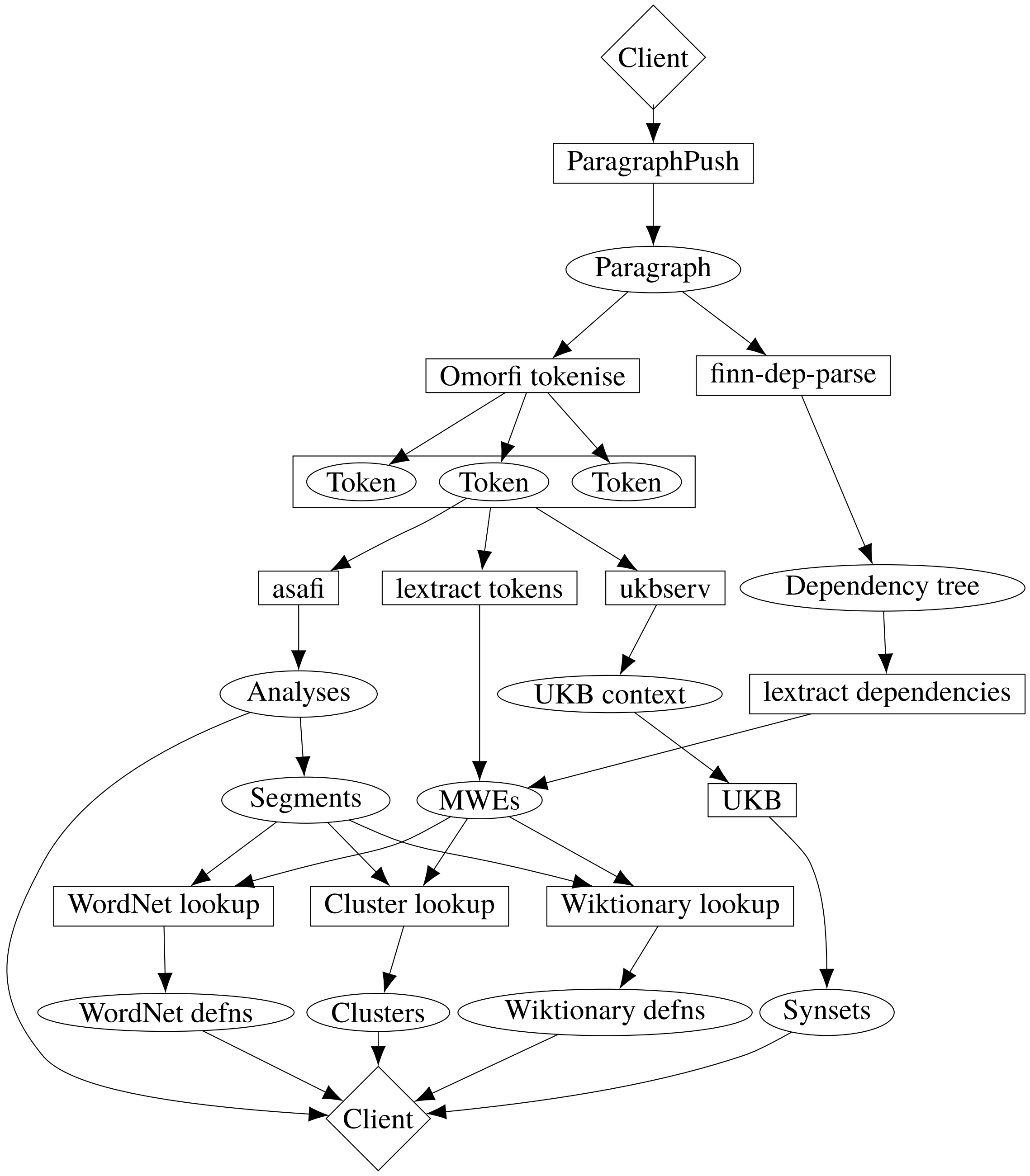


Figure 34: Diagram showing the runtime server side data flow of linguistic information to the TheWhatNow?! client. Elliptical nodes indicate some piece of data. Rectangles indicate data processing steps. The client indicated as a diamond.

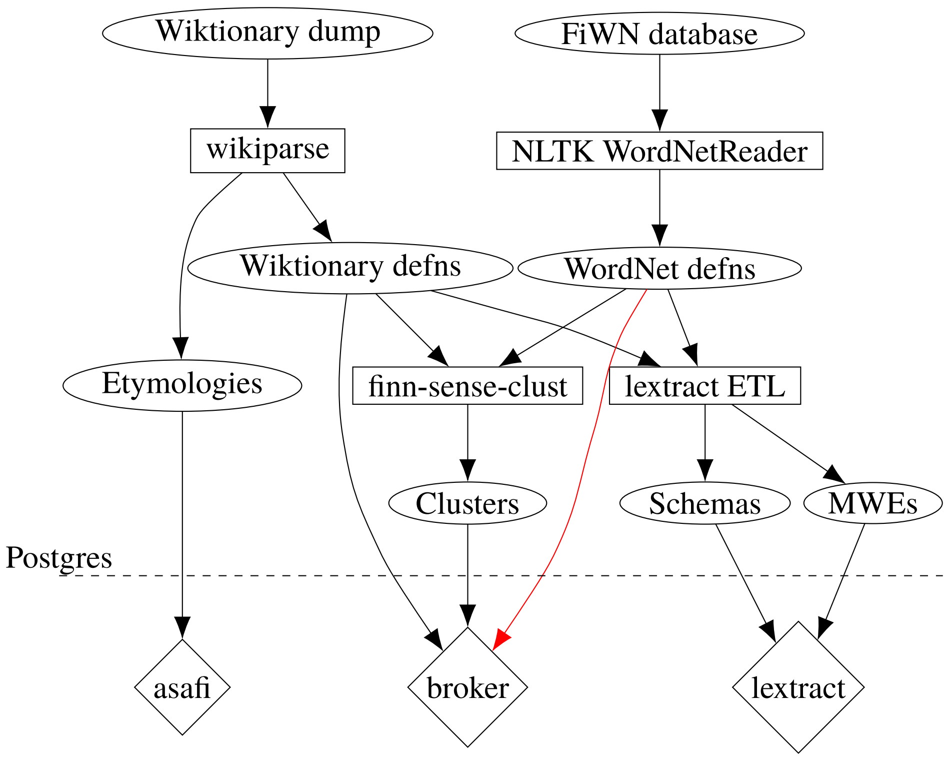


Figure 35: Diagram showing the ETL data flow of linguistic data for usage by TheWhatNow?! Elliptical nodes indicate some piece of data. Rectangles indicate data processing steps. Diamonds indicate downstream, non-ETL processes. Most of the data is loaded into Postgres before usage by the downstream processes, however the red line indicates this is not the case for FinnWordNet (FiWN) definitions, which are loaded directly from the PWN format database files.

formed eagerly, while analytical segmentation and definition and cluster lookup is performed on demand. This may seem impossible since UKB requires lemmas for the whole paragraph context, which would necessitate running asafi for this whole context in advance. To work around this, there is the ukbserv component which creates contexts for UKB by performing fixed point Omorfi extraction as described on page 47. Note that it is not always necessary to block until all of the eagerly executed tasks are complete before sending some kind of response to the client. For example, if a `CursorAnalysisRequest` comes when a parse tree from Finnish-dep-parser is already available, it is used together with lextract for MWE extraction, otherwise, the procedure which works on tokenised text is used first, and the dependency tree procedure deferred until the response from Finnish-dep-parser is ready. This approach allows for lower latency since we are not forced into a situation where we must always go linearly through the entire pipeline of linguistic analyses.

The ETL layer, shown in more detail in Figure 35 consists of first of running wikiparse, described in Section 5.1 so that Wiktionary definitions are available in the Postgres database. The word formation etymologies are then ready for use by asafi, described in Section 6.2. Wiktionary definitions and WordNet definitions are used together by to create sense clusters by the finn-sense-clust sense clustering component, described in Section 5.3, and to create indices for extracting MWEs by lextract's ETL component, described in Section 5.2.

7 Conclusion

This thesis began with an idea: Finnish language learners would be able to benefit from extended reading more readily and more extensively given in context help. From this idea, the thesis has followed a path of constructing and evaluating resources aiming to offer help focussed on word meaning and formation.

This conclusion first summarises the practical contributions made as part of this thesis in the form of software and language resources. Next, research contributions and findings are summarised. We finish by considering some possibilities for future work.

7.1 Software and language resource contributions

The following open source software resources were created as part of this thesis:

- STIFF: `https://github.com/frankier/STIFF`
  The main software package behind Chapter 3. It automatically creates the STIFF sense tagged corpus of Finnish. It includes and corpus stream format conversion tools to convert STIFF, EuroSense and the manually annotated corpus into a format usable by the Word Sense Disambiguation (WSD) tools as well as evaluation and plotting code.
- finntk: `https://github.com/frankier/finntk`
  Simple, high level toolkit for Finnish Natural Language Processing (NLP), mainly providing convenience methods for, and gluing together other tools. It can, for example, process information from Omorfi, FinnPOS, Wiktionary and FinnWordNet into comparable forms. The cores of some of the WSD techniques used are finn-wsd-eval is implemented here.
- expcomb: `https://github.com/frankier/expcomb`
  Python library with support code for evaluations which compare many combinations of experiments. It includes LaTeX table generation code and code to perform bootstrapping. The library integrates with SnakeMake (Köster and Rahmann 2012). It is used by finn-wsd-eval and finn-sense-clust.

- finn-wsd-eval: `https://github.com/frankier/finn-wsd-eval`
  The main software package behind Chapter 4. It contains code to apply expcomb and to set up the different WSD systems such as fetching any resources needed and converting between different result formats. Ultimately the process is automated to the extent that the whole evaluation can be rerun by simply running a command in a Docker container.
- wikiparse: `https://github.com/frankier/wikiparse`
  The main software package behind Section 5.1. It scrapes Wiktionary into structured data for use by lextract, finn-sense-clust and TheWhatNow?!
- lextract: `https://github.com/frankier/lextract`
  A software package developed in Section 5.2 to extract schemas and Multi-Word Expressions (MWEs) from Finnish text.
- finn-sense-clust: `https://github.com/frankier/finn-sense-clust`
  The main software package behind Section 5.3. It contains code to apply expcomb to evaluating different sense clustering methods on FinnWordNet (FiWN) and Wiktionary. It includes the implementation of all clustering techniques, as well as code to build the automatic evaluation data set and the manual evaluation data.
- asafi: `https://gitlab.com/frankier/asafi/`
  The main software package behind Section 6.2. It provides the type of word analyses required by TheWhatNow?!
- STIFF-explore: `https://github.com/frankier/STIFF-explore`
  Exploratory coding and plotting code related to this thesis.

Plugins and forks were made to the following open source software, and an attempt was made to contribute them back to the original authors where possible:

- extjwnl_fiwn: `https://github.com/frankier/extjwnl_fiwn`
  Java code to make extjwnl [1] interoperate with FiWN, so that SupWSD can be used together with FiWN.
- AutoExtend: `https://github.com/frankier/AutoExtend`
  AutoExtend fork to support FiWN and ConceptNet Numberbatch, used in sections 4.3.4 & 5.3.6.

1. Available at `https://github.com/extjwnl/extjwnl`.

- babelnet-lookup: `https://github.com/frankier/babelnet-lookup`
  babelnet-lookup fork to obtain the BabelNet-WordNet mapping used on page 56.
- FinnWordNet: `https://github.com/frankier/fiwn`
  A fork of FinnWordNet 2.0 made to fix the database so that it can interoperate with other tools. Other changes include the addition of the synthetic counts of Section 3.2.2, as well as adding extra data to make it easy to map between Princeton WordNet (PWN) and FiWN synsets.
- EuroSense: `https://github.com/frankier/eurosense`
  This repository contains a script attempting to fix EuroSense as described on page 54. In particular, the version of EuroSense obtained from `http://lcl.uniroma1.it/eurosense/` has the incorrect language associated with some annotations. The fixed corpus, which is the one described in Section 3.4 is also made available.
- aho-corasick: `https://github.com/frankier/pyahocorasick/commits/stiff-2018-09-20-3`
  This fork adds word/vocabulary based Aho-Corasick automata for extracting Finnish MWEs in Section 3.2. The original library could only operate with character based automata.

The following language resources were created:

- Sense annotated corpora for Finnish:
  - The STIFF corpus created in Chapter 3. The high recall and high precision variants are available at `https://archive.org/details/stiff-br4-bp4`.
  - The fixed high precision EuroSense corpus is available at `https://archive.org/details/eurosense-hp.fixed.xml`.
  - The manual corpus created in Section 3.5 is available from `https://github.com/frankier/finn-man-ann`.
- A Context2Vec model (see Section 4.3.7) for Finnish, made available at `https://archive.org/details/ctx2vec-b100-3epoch`.
- An extracted version of Finnish definitions from English Wiktionary created in Section 5.1 made available as an SQLite database at `https://github.com/frankier/wikiparse/releases/download/all190406to1223/`.

- Clustered and aligned word sense resources for Finnish:
  - The automatically created verb cluster evaluation data created in Section 5.3.2, which is available at `https://github.com/frankier/finn-sense-clust/releases/download/downloads/eval.tar.gz`.
  - The manually created noun cluster evaluation data also described in Section 5.3.3, which is available at `https://github.com/frankier/finn-sense-clust/tree/master/manclus`.
  - Sense clusters resulting from the *Sb+Ety+Ex* system, chosen in Section 5.3.6, distributed as part of the aforementioned Wiktionary SQLite database.
- A list of MWEs extracted from Wiktionary and FiWN in Section 5.2, which is also distributed as part of the aforementioned Wiktionary SQLite database.

Additionally, the TheWhatNow?! web application and browser extension were created and are made available at `https://thewhatnow.fi/`.

7.2 Research contributions

During the course of creating this thesis, the following was found in relation to trying to replicate and utilise previous work:

- OpenSubtitles2018 contains encoding errors in $\sim$1% of its Mandarin text. See page 44.
- According to an evaluation conducted as part of this thesis, EuroSense appears to have much lower precision and recall for Finnish as compared to the authors' own evaluation for other languages. See Figure 13 on page 65.
- Both Lesk++ and SupWSD produced results that were significantly worse than expected based on the English language evaluations of their authors. See Section 4.5.
- The sense clustering work of Snow et al. (2007) made inappropriate use of F_1-measure. See Section 5.3.4.

In terms of the results of evaluations performed as part of this thesis, the following is worth highlighting again:

- The STIFF corpus produced in Chapter 3 is competitive with at least one other automatically induced corpus, EuroSense, despite the simplicity of the technique used to produce it.
- The strongest WSD system according to Chapter 4 was UKB with frequency data — appearing to contradict previous results which show supervised systems outperforming knowledge based systems by some margin. However, there were promising initial results for using the manually created English training data of SemCor together with systems based on BERT.
- The similarity tuned BERT of Reimers and Gurevych (2019), used in Section 5.3 produces quite a strong system for clustering word senses based on definition alone.

7.3 Future work

More than once, this thesis breaks some new ground, but ultimately leaves quite a bit on the table. To some degree that is because it has been directed towards achieving a particular end goal for a particular application: TheWhatNow?! This results in quite a wide area in which it would be possible to go deeper. This section therefore concentrates only on the broadest and most obvious future directions.

In terms of creating a WSD system for Finnish, there is much that could be done. For example, the techniques of Scarlini, Pasini, and Navigli (2020), L. Huang et al. (2019) or Vial, Lecouteux, and Schwab (2019), which were published during the writing of this thesis, could be tried for Finnish. In terms of improving upon systems tried in the evaluation, the BERT system could be updated to use a more conventional final linear layer for feature selection rather than using nearest neighbour. Figure 19 showed that there is a need for more WSD evaluation data for Finnish, and making sure this is in place should be a priority for any future work in Finnish WSD.

In terms of sense clustering, reasonably strong results were obtained in Section 5.3 based on SentBert, and it may be interesting to try these approaches for English for comparison with previous work.

The analytical segmenter of Section 6.2 is rule based, and thus cannot handle out of vocabulary words. A machine learning approach such as that of Kann, Cotterell, and Schütze (2016) could be combined with the data developed here to address this.

For TheWhatNow?! the possible space of new features is also very large. We therefore consider possibilities which keep its scope to that of helping language comprehension with a "zoomed in" lexical focus on words and idioms. One direction would be to take into account learner vocabulary jointly with the most important words in the text to provide their definitions in line with the original text automatically, or produce a word list from a document for the reader to learn beforehand. TheWhatNow?! could be applied to new media such as the subtitles of a film. In terms of defining words, currently, this is done with English translations or definitions, but there are other possibilities such as Finnish definitions, showing related words such as synonyms and antonyms showing the word in additional disambiguating contexts, or making use of text simplification. The WSD task would then be extended to including deciding which type of word meaning hint to show based on both the context of the word and the learner's knowledge level.

Ideally TheWhatNow?! would take account of users' intentions and energy levels to direct its reading assistance. For example, a reader might in some cases be reading the text in order to achieve some external task and therefore have less energy for analysing the text or incidental learning. In this case the best strategy might be to present chunks of translated text beyond single words in line with the text without further linguistic information to minimise distraction. Other times a learner could be reading with the particular intention of studying the language. In this case, features like extracting word lists and displaying morphological analyses and usage notes become more relevant. Making a tool automatically choose the correct settings depending on the user's state like this is an open problem, but a starting point here could be to make use of how goal oriented the text itself appears.

The question of whether systems such as TheWhatNow?! truly help language learners is a pertinent one. To minimise confounders, and thus obtain valid data, any experiment is likely to have limited scope. Further research into this direction including qualitative and quantitative user evaluations to validate existing features and point to new ones is thus an important piece of future work.

Bibliography

Agirre, Eneko, Oier Lopez de Lacalle, and Aitor Soroa. 2014. ”Random walks for knowledge-based word sense disambiguation”. *Computational Linguistics* 40 (1): 57–84. doi:10.1162/COLI_a_00164.

Agirre, Eneko, Oier López de Lacalle, and Aitor Soroa. 2018. ”The risk of sub-optimal use of Open Source NLP Software: UKB is inadvertently state-of-the-art in knowledge-based WSD”. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS),* 29–33. Melbourne, Australia: Association for Computational Linguistics. doi:10.18653/v1/W18-2505. arXiv: 1805.04277.

Aho, Alfred V., and Margaret J. Corasick. 1975. ”Efficient string matching: an aid to bibliographic search”. *Communications of the ACM* 18 (6): 333–340. ISSN: 00010782. doi:10.1145/360825.360855.

Aitchison, J. 2012. *Words in the mind: An introduction to the mental lexicon.* 4th. New York, NY: John Wiley & Sons. ISBN: 9780470656471.

Amdahl, Gene M. 1967. ”Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities”. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference,* 483–485. AFIPS ’67 (Spring). Atlantic City, New Jersey: ACM. doi:10.1145/1465482.1465560.

Arora, Sanjeev, Yingyu Liang, and Tengyu Ma. 2017. ”A Simple but Tough-to-Beat Baseline for Sentence Embeddings”. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017.* https://openreview.net/forum?id=SyK00v5xx.

Baker, Collin F, Charles J Fillmore, and John B Lowe. 1998. ”The berkeley framenet project”. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1,* 86–90. Association for Computational Linguistics. doi:10.3115/980451.980860.

Banerjee, Satanjeev, and T Pedersen. 2002. ”Adapting the Lesk algorithm for word sense disambiguation to WordNet”. Master’s thesis, University of Minnesota. doi:10.1007/3-540-45715-1_11.

Banerjee, Satanjeev, and Ted Pedersen. 2003. ”Extended Gloss Overlaps As a Measure of Semantic Relatedness”. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence,* 3:805–810. IJCAI'03. Acapulco, Mexico: Morgan Kaufmann Publishers Inc. doi:10.5555/1630659.1630775.

Basile, Pierpaolo, Annalina Caputo, and Giovanni Semeraro. 2014. ”An Enhanced Lesk Word Sense Disambiguation Algorithm through a Distributional Semantic Model.” In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers,* 1591–1600. http://www.aclweb.org/anthology/C14-1151.

Berg-Kirkpatrick, Taylor, David Burkett, and Dan Klein. 2012. ”An Empirical Investigation of Statistical Significance in NLP”. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning,* 995–1005. Jeju Island, Korea: Association for Computational Linguistics. https://www.aclweb.org/anthology/D12-1091.

Bhalla, Vishal, and Klara Klimcikova. 2019. ”Evaluation of automatic collocation extraction methods for language learning”. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications,* 264–274. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/W19-4428.

Biemann, Chris. 2006. ”Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems”. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing,* 73–80. TextGraphs-1. Stroudsburg, PA, USA: Association for Computational Linguistics. doi:10.3115/1654758.1654774.

———. 2011. *Structure discovery in natural language.* Springer Science & Business Media. doi:10.1007/978-3-642-25923-4.

Bleyhl, Werner. 2009. ”The hidden paradox of foreign language instruction. Or: Which are the real foreign language learning processes”. *Input Matters in SLA. Clevedon: Multilingual Matters:* 137–55. doi:10.21832/9781847691118-010.

Bond, Francis, and Ryan Foster. 2013. ”Linking and Extending an Open Multilingual Wordnet”. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 1352–1362. Sofia, Bulgaria: Association for Computational Linguistics. `https://www.aclweb.org/anthology/P13-1133`.

Bond, Francis, and Kyonghee Paik. 2012. ”A Survey of WordNets and their Licenses”. In *Proceedings of the 6th Global WordNet Conference (GWC 2012),* 64–71. December. Tribun EU. ISBN: 9788026302445. `http://web.mysites.ntu.edu.sg/fcbond/open/pubs/2012-gwc-wn-license.pdf`.

Booij, G. 1996. ”Inherent versus contextual inflection and the split morphology hypothesis”. In *Yearbook of morphology 1995,* 1–16. Dordrecht, Netherlands. doi:`10.1007/978-94-017-3716-6_1`.

Borin, Lars, Markus Forsberg, and Lennart Lönngren. 2013. ”SALDO: a touch of yin to WordNet's yang”. *Language Resources and Evaluation* 47 (4): 1191–1211. ISSN: 1574-0218. doi:`10.1007/s10579-013-9233-4`.

Boulton, Alex. 2009. ”Data-driven Learning: Reasonable Fears and Rational Reassurance”. *Indian Journal of Applied Linguistics* 35 (1): 81–106. ISSN: 0379-0037. `https://hal.archives-ouvertes.fr/hal-00326990`.

Bovi, Claudio Delli, Jose Camacho-Collados, Alessandro Raganato, and Roberto Navigli. 2017. ”Eurosense: Automatic harvesting of multilingual sense annotations from parallel text”. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* 2:594–600. doi:`10.18653/v1/P17-2094`.

Brown, Peter F, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. ”The mathematics of statistical machine translation: Parameter estimation”. *Computational linguistics* 19 (2): 263–311. `https://www.aclweb.org/anthology/J93-2003`.

Callison-Burch, Chris, Miles Osborne, and Philipp Koehn. 2006. ”Re-evaluating the Role of Bleu in Machine Translation Research”. In *11th Conference of the European Chapter of the Association for Computational Linguistics.* Trento, Italy: Association for Computational Linguistics. `https://www.aclweb.org/anthology/E06-1032`.

Chan, Yee Seng, and Hwee Tou Ng. 2005. ”Scaling up Word Sense Disambiguation via Parallel Texts”. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3,* 1037–1042. AAAI’05. Pittsburgh, Pennsylvania: AAAI Press. ISBN: 157735236x. https://dl.acm.org/doi/10.5555/1619499.1619500.

Chenu, Florence, and Harriet Jisa. 2009. ”Reviewing some similarities and differences in L1 and L2 lexical development”. *Acquisition et interaction en langue étrangère,* number Aile... Lia 1: 17–38. http://journals.openedition.org/aile/4506.

Conneau, Alexis, Guillaume Lample, Marc’Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. ”Word Translation Without Parallel Data”. In *International Conference on Learning Representations.* arXiv: 1710.04087. https://openreview.net/forum?id=H196sainb.

Creutz, Mathias, and Krista Lagus. 2005. ”Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0”. In *Publications in Computer and Information Science, Report A,* 81:1–27. Helsinki University of Technology. https://tuhat.helsinki.fi/ws/portalfiles/portal/62462066/Creutz05tr.pdf.

De Smit, Merlijn. 2006. *Language contact and structural change: An Old Finnish case study.* Studia Fennica Stockholmiensia. Acta Universitatis Stockholmiensis. ISBN: 9185445533.

DeFrancis, John. 1984. ”The Chinese language: Fact and fantasy”. *Language* 62 (3): 346. ISSN: 1535-0665.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. ”BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers),* 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics. doi:10.18653/v1/N19-1423.

Dupuy, Beatrice, and Stephen D. Krashen. 1998. ”From Lower-Division to Upper-Division Foreign Language Classes”. *ITL - International Journal of Applied Linguistics* 119-120 (1): 1–7. ISSN: 0019-0829. doi:10.1075/itl.119-120.01dup.

Fares, Murhaf, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. 2017. ”Word vectors, reuse, and replicability: Towards a community repository of large-text resources”. In *Proceedings of the 21st Nordic Conference on Computational Linguistics,* 271–276. Gothenburg, Sweden: Association for Computational Linguistics. `https://www.aclweb.org/anthology/W17-0237`.

Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database,* 71:423. 3. MIT Press. ISBN: 026206197X. doi:`10.1139/h11-025`.

Firth, J. 1957. ”A Synopsis of Linguistic Theory 1930-1955”. In *Studies in Linguistic Analysis.* Reprinted in Palmer, F. (ed. 1968) Selected Papers of J. R. Firth, Longman, Harlow. Philological Society, Oxford.

Fitzgerald, Alannah, Shaoqun Wu, and María José Marín. 2015. ”FLAX: Flexible and open corpus-based language collections development”. *10 years of the LLAS elearning symposium: case studies in good practice:* 215–227. doi:`10.14705/rpnet.2015.000281`.

Frey, Brendan J., and Delbert Dueck. 2007. ”Clustering by Passing Messages Between Data Points”. *Science* 315 (5814): 972–976. ISSN: 0036-8075. doi:`10.1126/science.1136800`.

Galdi, Paola, Francesco Napolitano, and Roberto Tagliaferri. 2014. ”A comparison between Affinity Propagation and assessment based methods in finding the best number of clusters”. In *Proceedings of CIBB.*

Gale, William A., and Kenneth W. Church. 1993. ”A Program for Aligning Sentences in Bilingual Corpora”. *Computational Linguistics* 19 (1): 75–102. `https://www.aclweb.org/anthology/J93-1004`.

Gale, William, Kenneth Ward Church, and David Yarowsky. 1992. ”Estimating Upper and Lower Bounds on the Performance of Word-sense Disambiguation Programs”. In *Proceedings of the 30th Annual Meeting on Association for Computational Linguistics,* 249–256. ACL ’92. Newark, Delaware: Association for Computational Linguistics. doi:`10.3115/981967.981999`.

Godwin-Jones, Robert. 2018. ”Contextualized vocabulary learning”. *Language Learning & Technology* 22 (3): 1–19. doi:10125/44651.

Graën, Johannes, and Gerold Schneider. 2017. ”Crossing the border twice: Reimporting prepositions to alleviate L1-specific transfer errors”. In *Proceedings of the joint workshop on NLP for Computer Assisted Language Learning and NLP for Language Acquisition,* 18–26. Gothenburg, Sweden: LiU Electronic Press. doi:10.5167/uzh-137099.

Häkkinen, Kaisa. 1992. ”Suomen perussanaston etymologiset kerrostumat [The etymological strata of the Finnish lexicon]”. Document in Finnish, *Virittäjä* 96 (1). https://journal.fi/virittaja/article/view/38498.

———. 1997. ”Kuinka ruotsin kieli on vaikuttanut suomeen? [How has the Swedish language influenced Finnish?]” Document in Finnish, *Sananjalka: Suomen kielen seuran vuosikirja:* 31–53. doi:10.30673/sja.86585.

Halpern, Jack, and Jouni Kerman. 1999. ”Pitfalls and Complexities of Chinese to Chinese Conversion”. In *International Unicode Conference (14th) in Boston.* http://www.mt-archive.info/MTS-1999-Halpern.pdf.

Hansen, Björn, and Ferdinand De Haan. 2009. ”Modal verbs in Balto-Finnic”. In *Modals in the languages of Europe: A reference work,* 363–402. Walter de Gruyter. doi:10.1515/9783110219210.3.363.

Hassan, Hany, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, et al. 2018. ”Achieving Human Parity on Automatic Chinese to English News Translation”. arXiv: 1803.05567.

Haverinen, Katri, Jenna Kanerva, Samuel Kohonen, Anna Missilä, Stina Ojala, Timo Viljanen, Veronika Laippala, and Filip Ginter. 2015. ”The Finnish proposition bank”. *Language Resources and Evaluation* 49 (4): 907–926. doi:10.1007/s10579-015-9310-y.

Heaps, H. S. 1978. *Information retrieval, computational and theoretical aspects.* Academic Press. ISBN: 0123357500. https://archive.org/details/informationretri0000heap.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory". *Neural Computation* 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735.

Hovy, Eduard, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. "OntoNotes: The 90% Solution". In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers,* 57–60. New York City, USA: Association for Computational Linguistics. doi:10.3115/1614049.1614064.

Huang, Chu-Ren, Shu-Kai Hsieh, Jia-Fei Hong, Yun-Zhu Chen, I-Li Su, Yong-Xiang Chen, and Sheng-Wei Huang. 2010. "Chinese WordNet: Design and Implementation of a cross-lingual knowledge processing infrastructure". (Document in Chinese), *Journal of Chinese Information Processing* 24 (2): 14–23. http://jcip.cipsc.org.cn/EN/abstract/abstract1340.shtml.

Huang, Luyao, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. "GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge". In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 3509–3514. Hong Kong, China: Association for Computational Linguistics. doi:10.18653/v1/D19-1355.

Hutchins, John. 1995. "The whisky was invisible" or persistent myths of MT". *MT News International* 11:17–18. http://www.hutchinsweb.me.uk/MTNI-11-1995.pdf.

Hyvärinen, Irma. 2019. "Compounds and multi-word expressions in Finnish: Compounds and Multi-Word Expressions", 307–336. ISBN: 9783110632446. doi:10.1515/9783110632446-011.

Iacobacci, Ignacio, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. "Embeddings for Word Sense Disambiguation: An Evaluation Study". In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 897–907. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/P16-1085.

Jaccard, Paul. 1926. *Le coefficient generique et le coefficient de communaute dans la flore marocaine.* Impr. Commerciale.

Järvinen, Pertti. 2012. *On research methods.* Tampereen yliopistopaino. ISBN: 9789529923342.

Jurafsky, Dan, and James H Martin. 2019a. ”19. Word Senses and WordNet”. In 3rd ed. draft of Speech and language processing. https://web.stanford.edu/~jurafsky/slp3/.

———. 2019b. ”6. Vector Semantics and Embeddings”. In 3rd ed. draft of Speech and language processing. https://web.stanford.edu/~jurafsky/slp3/.

Kallio, Petri. 2012. ”The Prehistoric Germanic Loanword Strata in Finnic”. In *A Linguistic Map of Prehistoric Northern Europe,* 225–238. ISBN: 978-9525667424. https://www.sgr.fi/sust/sust266/sust266_kallio.pdf.

Kann, Katharina, Ryan Cotterell, and Hinrich Schütze. 2016. ”Neural Morphological Analysis: Encoding-Decoding Canonical Segments”. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* 961–967. Austin, Texas: Association for Computational Linguistics. doi:10.18653/v1/D16-1097.

Karlsson, Fred. 2015. *Finnish: an essential grammar.* 343. ISBN: 1138821586. doi:10.4324/9781315743233.

Kilgarriff, Adam, Vít Baisa, Jan Busta, Milos Jakubícek, Vojtech Kovár, Jan Michelfeit, Pavel Rychlý, and Vít Suchomel. 2014. ”The Sketch Engine: ten years on”. *Lexicography* 1 (1): 7–36. doi:10.1007/s40607-014-0009-9.

Kilgarriff, Adam, Fredrik Marcowitz, Simon Smith, and James Thomas. 2015. ”Corpora and language learning with the Sketch Engine and SKELL”. *Revue française de linguistique appliquée* 20 (1): 61–80. doi:10.3917/rfla.201.0061.

Kilgarriff, Adam, and Martha Palmer. 2000. ”Introduction to the special issue on SENSEVAL”. *Computers and the Humanities* 34 (1-2): 1–13. doi:10.1023/A%3A1002619001915.

Kilgarriff, Adam, and Joseph Rosenzweig. 2000a. ”English Senseval: Report and Results”. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC’00).* Athens, Greece: European Language Resources Association (ELRA). https://www.aclweb.org/anthology/L00-1005.

Kilgarriff, Adam, and Joseph Rosenzweig. 2000b. ”Framework and results for English SENSEVAL”. *Computers and the Humanities* 34 (1): 15–48. doi:10.1023/A%3A1002693207386.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. ”Moses: Open source toolkit for statistical machine translation”. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions,* 177–180. Association for Computational Linguistics. doi:10.3115/1557769.1557821.

Köster, Johannes, and Sven Rahmann. 2012. ”Snakemake—a scalable bioinformatics workflow engine”. *Bioinformatics* 28 (19): 2520–2522. ISSN: 1367-4803. doi:10.1093/bioinformatics/bts480. eprint: http://oup.prod.sis.lan/bioinformatics/article-pdf/28/19/2520/819790/bts480.pdf.

Krashen, Stephen. 1989. ”We Acquire Vocabulary and Spelling by Reading: Additional Evidence for the Input Hypothesis”, 73 (4): 440–464. ISSN: 15404781. doi:10.1111/j.1540-4781.1989.tb05325.x.

Krashen, Stephen D. 1982. *Principles and Practice in Second Language Acquisition.* http://www.sdkrashen.com/content/books/principles_and_practice.pdf.

Krashen, Stephen D., and Tracy D. Terrell. 1998. *The Natural Approach.* http://www.sdkrashen.com/content/books/the_natural_approach.pdf.

Kusner, Matt, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. ”From word embeddings to document distances”. In *International conference on machine learning,* 957–966.

Lacalle, Maddalen Lopez de, Egoitz Laparra, Itziar Aldabe, and German Rigau. 2016. ”A Multilingual Predicate Matrix”. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16),* 2662–2668. Portorož, Slovenia: European Language Resources Association (ELRA). https://www.aclweb.org/anthology/L16-1423.

Laippala, Veronika, and Filip Ginter. 2014. ”Syntactic n-gram collection from a large-scale corpus of internet finnish”. In *Human Language Technologies-The Baltic Perspective: Proceedings of the Sixth International Conference Baltic HLT,* 268:184.

Lesk, Michael. 1986. ”Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone”. In *Proceedings of the 5th Annual International Conference on Systems Documentation,* 24–26. SIGDOC ’86. Toronto, Ontario, Canada: ACM. ISBN: 0-89791-224-1. doi:10.1145/318723.318728.

Lightbown, P. M., and N. Spada. 2013. *How languages are learned.* 3rd ed, edited by Nina Spada, 249. Oxford handbooks for language teachers. Oxford: Oxford University Press. ISBN: 0194370003. doi:10.1017/CBO9781107415324.004.

Lindén, Krister, and Lauri Carlson. 2010. ”FinnWordNet–Finnish WordNet by Translation”. *LexicoNordica–Nordic Journal of Lexicography* 17:119–140. http://www.ling.helsinki.fi/~klinden/pubs/FinnWordnetInLexicoNordica-en.pdf.

Lison, Pierre, Jörg Tiedemann, and Milen Kouylekov. 2018. ”OpenSubtitles2018: Statistical Rescoring of Sentence Alignments in Large, Noisy Parallel Corpora”. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).* Miyazaki, Japan: European Language Resources Association (ELRA). https://www.aclweb.org/anthology/L18-1275.

Lyons, J. 1968. *Introduction to Theoretical Linguistics.* Cambridge, England: Cambridge University Press. doi:10.1017/CBO9781139165570.

Maaten, L.J.P. van der, E. O. Postma, and H. Jaap van den Herik. 2007. ”Dimensionality Reduction: A Comparative Review”.

MacWhinney, Brian. 2008. ”A unified model”.

Mallery, John C. 1988. ”Thinking About Foreign Policy: Finding an Appropriate Role for Artificially Intelligent Computers”. Master’s thesis, Massachusetts Institute of Technology.

Manion, Steve L. 2015. ”SUDOKU: Treating Word Sense Disambiguation & Entitiy Linking as a Deterministic Problem-via an Unsupervised & Iterative Approach”. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015),* 365–369. Denver, Colorado: Association for Computational Linguistics. doi:10.18653/v1/S15-2062.

Manning, Christopher D, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval.* Volume 1. 1. Cambridge university press. doi:10.1017/CBO9780511809071. https://nlp.stanford.edu/IR-book/.

Matthews, P. H. 1991. *Morphology.* Cambridge Textbooks in Linguistics. Cambridge, England: Cambridge University Press. doi:10.1017/CBO9781139166485.

———, editor. 2007. *The concise Oxford dictionary of linguistics.* Oxford, England: Oxford University Press.

Mcquillan, Jeff. 1997. ”Does anyone finish the Berlitz tapes? A novel measure of perseverance for commercial language courses”. http://backseatlinguist.com/blog/wp-content/uploads/2012/01/Berlitz%7B%5C_%7DTape.pdf.

Melamud, Oren, Jacob Goldberger, and Ido Dagan. 2016. ”context2vec: Learning Generic Context Embedding with Bidirectional LSTM”. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning,* 51–61. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/K16-1006.

Meurers, Detmar, Ramon Ziai, Luiz Amaral, Adriane Boyd, Aleksandar Dimitrov, Vanessa Metcalf, and Niels Ott. 2010. ”Enhancing authentic web pages for language learners”. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications,* 10–18. Association for Computational Linguistics. http://www.aclweb.org/anthology/W/W10/W10-1002.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. ”Efficient Estimation of Word Representations in Vector Space”. In *Proceedings of the First International Conference on Learning Representations,* volume abs/1301.3781, 1–13. Scottsdale, Arizona, USA. arXiv: 1301.3781.

Miller, George A. 1995. ”WordNet: A Lexical Database for English”. *Communications of the ACM* (New York, NY, USA) 38 (11): 39–41. ISSN: 0001-0782. doi:10.1145/219717.219748.

Miller, George A., Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. ”Using a Semantic Concordance for Sense Identification”. In *Proceedings of the Workshop on Human Language Technology,* 240–243. HLT ’94. Plainsboro, NJ: Association for Computational Linguistics. ISBN: 1558603573. doi:10.3115/1075812.1075866.

Miller, George A, Claudia Leacock, Randee Tengi, and Ross T Bunker. 1993. ”A semantic concordance”. In *Proceedings of the workshop on Human Language Technology,* 303–308. Association for Computational Linguistics.

Moirón, Begoña Villada, and Jörg Tiedemann. 2006. ”Identifying idiomatic expressions using automatic word-alignment”. In *Proceedings of the Workshop on Multi-word-expressions in a multilingual context.* https://www.aclweb.org/anthology/W06-2405.

Moro, Andrea, and Roberto Navigli. 2015. ”SemEval-2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking”. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015),* 288–297. Denver, Colorado: Association for Computational Linguistics. doi:10.18653/v1/S15-2049.

Moro, Andrea, Alessandro Raganato, and Roberto Navigli. 2014. ”Entity Linking meets Word Sense Disambiguation: a Unified Approach”. *Transactions of the Association for Computational Linguistics* 2:231–244. doi:10.1162/tacl_a_00179.

Navigli, Roberto. 2006. ”Meaningful Clustering of Senses Helps Boost Word Sense Disambiguation Performance”. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics,* 105–112. Sydney, Australia: Association for Computational Linguistics. doi:10.3115/1220175.1220189.

———. 2009. ”Word Sense Disambiguation: A Survey”. *ACM Computing Surveys* (New York, NY, USA) 41 (2): 10:1–10:69. ISSN: 0360-0300. doi:10.1145/1459352.1459355.

Navigli, Roberto, and Mirella Lapata. 2010. ”An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation”. *IEEE transactions on pattern analysis and machine intelligence* (Washington, DC, USA) 32 (4): 678–692. ISSN: 0162-8828. doi:`10.1109/TPAMI.2009.36`.

Navigli, Roberto, and Simone Paolo Ponzetto. 2012. ”BabelNet: The Automatic Construction, Evaluation and Application of a Wide-coverage Multilingual Semantic Network”. *Artificial Intelligence* (Essex, UK) 193:217–250. ISSN: 0004-3702. doi:`10.1016/j.artint.2012.07.001`.

Nerbonne, J., D. Dokter, and P. Smit. 1998. ”Morphological processing and computer-assisted language learning”. *Computer Assisted Language Learning* 11 (5): 543–559. ISSN: 0958-8221. doi:`10.1076/call.11.5.543.5660`.

Nielson, Katharine B. 2011. ”Self-Study with Language Learning Software in the Workplace: What Happens?” *Language Learning & Technology* 15 (3): 110–129. ISSN: 1094-3501. `http://llt.msu.edu/issues/october2011/nielson.pdf`.

Novotný, Vít. 2018. ”Implementation Notes for the Soft Cosine Measure”. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management - CIKM ’18*. doi:`10.1145/3269206.3269317`.

Och, Franz. 2006. *Statistical machine translation live.* `https://ai.googleblog.com/2006/04/statistical-machine-translation-live.html`.

Och, Franz Josef, and Hermann Ney. 2003. ”A Systematic Comparison of Various Statistical Alignment Models”. *Computational Linguistics* 29 (1): 19–51. doi:`10.1162/089120103321337421`.

Oele, Dieke, and Gertjan van Noord. 2017. ”Distributional Lesk: Effective Knowledge-Based Word Sense Disambiguation”. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers.* `https://www.aclweb.org/anthology/W17-6931`.

Osimo, Bruno. 2008. ”Meaning in translation: A model based on translation shifts”. *Linguistica Antverpiensia, New series–Themes in translation studies,* number 7.

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab. `http://ilpubs.stanford.edu:8090/422/`.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. "The Proposition Bank: An Annotated Corpus of Semantic Roles". *Computational Linguistics* 31 (1): 71–106. doi:`10.1162/0891201053630264`.

Panchenko, Alexander, Eugen Ruppert, Stefano Faralli, Simone Paolo Ponzetto, and Chris Biemann. 2017. "Unsupervised Does Not Mean Uninterpretable: The Case for Word Sense Induction and Disambiguation". In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers,* 86–98. Valencia, Spain: Association for Computational Linguistics.

Papandrea, Simone, Alessandro Raganato, and Claudio Delli Bovi. 2017. "SUPWSD: A Flexible Toolkit for Supervised Word Sense Disambiguation". In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations,* 103–108.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. "Bleu: a Method for Automatic Evaluation of Machine Translation". In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics,* 311–318. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics. doi:`10.3115/1073083.1073135`.

Pasini, Tommaso, and José Camacho-Collados. 2018. "A Short Survey on Sense-Annotated Corpora for Diverse Languages and Resources". *CoRR* abs/1802.04744. arXiv: `1802.04744`.

Pasini, Tommaso, and Roberto Navigli. 2017. "Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data". In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* 78–88. Copenhagen, Denmark: Association for Computational Linguistics. doi:`10.18653/v1/D17-1008`.

Passonneau, Rebecca J., Collin F. Baker, Christiane Fellbaum, and Nancy Ide. 2012. ”The MASC Word Sense Corpus”. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12),* 3025–3030. Istanbul, Turkey: European Language Resources Association (ELRA). `https://www.aclweb.org/anthology/L12-1335`.

Pearce, Sarah. 2016. ”Authentic learning: what, why and how?”, 2016 (April): 2. `http://www.acel.org.au/acel/ACEL_docs/Publications/e-Teaching/2016/e-Teaching_2016_10.pdf`.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. ”Scikit-learn: Machine learning in Python”. *Journal of machine learning research* 12 (Oct): 2825–2830.

Pele, Ofir, and Michael Werman. 2009. ”Fast and robust earth mover's distances”. In *2009 IEEE 12th International Conference on Computer Vision,* 460–467. IEEE. doi:`10.1109/ICCV.2009.5459199`.

Pelevina, Maria, Nikolay Arefiev, Chris Biemann, and Alexander Panchenko. 2016. ”Making Sense of Word Embeddings”. In *Proceedings of the 1st Workshop on Representation Learning for NLP,* 174–183. Berlin, Germany: Association for Computational Linguistics. doi:`10.18653/v1/W16-1620`.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. 2014. ”Glove: Global Vectors for Word Representation”. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 1532–1543. Doha, Qatar: Association for Computational Linguistics. doi:`10.3115/v1/D14-1162`.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. ”Deep contextualized word representations”. doi:`10.18653/v1/N18-1202`.

Peters, Matthew E., Sebastian Ruder, and Noah A. Smith. 2019. ”To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks”. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019),* 7–14. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/W19-4302. https://www.aclweb.org/anthology/W19-4302.

Peterson, L. E. 2009. ”K-nearest neighbor”. Revision #136646, *Scholarpedia* 4 (2): 1883. doi:10.4249/scholarpedia.1883.

Piepho, Hans-Peter. 2004. ”An Algorithm for a Letter-Based Representation of All-Pairwise Comparisons”. *Journal of Computational and Graphical Statistics* 13 (2): 456–466. doi:10.1198/1061860043515.

Pirinen, Tommi A. 2015a. ”Development and Use of Computational Morphology of Finnish in the Open Source and Open Science Era: Notes on Experiences with Omorfi Development.” *SKY Journal of Linguistics* 28:381–393.

———. 2015b. ”Omorfi — Free and open source morphological lexical database for Finnish”. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015),* 313–315. Vilnius, Lithuania: Linköping University Electronic Press, Sweden. https://www.aclweb.org/anthology/W15-1844.

Pyysalo, Sampo, Jenna Kanerva, Anna Missilä, Veronika Laippala, and Filip Ginter. 2015. ”Universal Dependencies for Finnish”. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015),* 163–172. Vilnius, Lithuania: Linköping University Electronic Press, Sweden. https://www.aclweb.org/anthology/W15-1821.

Quillian, M. Ross. 1969. ”The Teachable Language Comprehender: A Simulation Program and Theory of Language”. *Communications of the ACM* (New York, NY, USA) 12 (8): 459–476. ISSN: 0001-0782. doi:10.1145/363196.363214.

Raganato, Alessandro, Jose Camacho-Collados, and Roberto Navigli. 2017. ”Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison”. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers,* 99–110. Valencia, Spain: Association for Computational Linguistics. doi:`10.18653/v1/E17-1010`.

Řehůřek, Radim, and Petr Sojka. 2010. ”Software Framework for Topic Modelling with Large Corpora”. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks,* 45–50. Valletta, Malta: ELRA. `http://is.muni.cz/publication/884893/en`.

Reimers, Nils, and Iryna Gurevych. 2019. ”Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 3982–3992. Hong Kong, China: Association for Computational Linguistics. doi:`10.18653/v1/D19-1410`.

Resnik, Philip. 2006. ”WSD in NLP Applications”. In *Word Sense Disambiguation: Algorithms and Applications,* 300–302. doi:`10.1007/978-1-4020-4809-8_11`.

Reynolds, Robert, Eduard Schaf, and Detmar Meurers. 2014. ”A VIEW of Russian: Visual Input Enhancement and Adaptive Feedback”. In *Proceedings of the third workshop on NLP for computer-assisted language learning,* 98–112. Uppsala, Sweden: LiU Electronic Press. `https://www.aclweb.org/anthology/W14-3508`.

Richard, G. 1999. ”A global perspective on bilingualism and bilingual education”. `http://www.cal.org/content/download/1803/19986/file/AGlobalPerspectiveonBilingualism.pdf`.

Robertson, Frankie. 2016. ”Morphological parsing with lexical transducers: a case study of OMorFi”. Bachelor’s Thesis. `http://urn.fi/URN:NBN:fi:jyu-201606012807`.

———. 2019. ”A Contrastive Evaluation of Word Sense Disambiguation Systems for Finnish”. In *Proceedings of the Fifth International Workshop on Computational Linguistics for Uralic Languages,* 42–54. Tartu, Estonia: Association for Computational Linguistics. `https://www.aclweb.org/anthology/W19-0304`.

Rothe, Sascha, and Hinrich Schütze. 2017. "Autoextend: Combiningword embeddings with semantic resources". *Computational Linguistics* 43 (3): 593–617. ISSN: 15309312. doi:10.1162/COLI_a_00294.

Rücklé, Andreas, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. "Concatenated p-mean Word Embeddings as Universal Cross-Lingual Sentence Representations". arXiv: 1803.01400.

Russell, Stuart Jonathan, and Peter Norvig. 2010. *Artificial intelligence: a modern approach.* 3rd edition. Upper Saddle River, N.J., Harlow: Pearson Education. ISBN: 0136042597.

Salton, Gerard, editor. 1971. *The SMART Retrieval System – Experiments in Automatic Document Processing.* Prentice-Hall, Inc. ISBN: 0138145253.

Scarlini, Bianca, Tommaso Pasini, and Roberto Navigli. 2020. "SensEmBERT: Context-Enhanced Sense Embeddings for Multilingual Word Sense Disambiguation". In *Proceedings of the Thirty-Fourth Conference on Artificial Intelligence.* Association for the Advancement of Artificial Intelligence. https://pasinit.github.io/papers/scarlini_etal_aaai2020.pdf.

Schmidt, Richard W. 1990. "The Role of Consciousness in Second Language Learning". *Applied Linguistics* 11 (2): 129–158. ISSN: 0142-6001. doi:10.1093/applin/11.2.129.

Schuler, Karin Kipper. 2006. "VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon". PhD thesis, University of Pennsylvania. http://verbs.colorado.edu/~kipper/Papers/dissertation.pdf.

Schütze, Hinrich. 1998. "Automatic Word Sense Discrimination". *Computational Linguistics* 24 (1): 97–123. https://www.aclweb.org/anthology/J98-1004.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. "Neural Machine Translation of Rare Words with Subword Units". In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 1715–1725. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/P16-1162.

Shapiro, Naomi Tachikawa. 2016. ”Splitting compounds with ngrams”. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,* 630–640. Osaka, Japan. https://www.aclweb.org/anthology/C16-1061.

Sharwood Smith, Michael A. 1993. ”Input Enhancement in Instructed SLA View project”. doi:10.1017/S0272263100011943.

Sidorov, Grigori, Alexander Gelbukh, Helena Gómez-Adorno, and David Pinto. 2014. ”Soft similarity and soft cosine measure: Similarity of features in vector space model”. *Computación y Sistemas* 18 (3): 491–504. doi:10.13053/cys-18-3-2043.

Silfverberg, Miikka, Teemu Ruokolainen, Krister Lindén, and Mikko Kurimo. 2016. ”FinnPos: an open-source morphological tagging and lemmatization toolkit for Finnish”. *Language Resources and Evaluation* 50 (4): 863–878. doi:10.1007/s10579-015-9326-3.

Snow, Rion, Sushant Prakash, Daniel Jurafsky, and Andrew Y. Ng. 2007. ”Learning to Merge Word Senses”. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL),* 1005–1014. Prague, Czech Republic: Association for Computational Linguistics. https://www.aclweb.org/anthology/D07-1107.

Speer, Robyn, Joshua Chin, and Catherine Havasi. 2017. ”ConceptNet 5.5: An Open Multilingual Graph of General Knowledge”. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence,* 4444–4451. AAAI’17. San Francisco, California, USA: AAAI Press. arXiv: 1612.03975. https://dl.acm.org/doi/10.5555/3298023.3298212.

Speer, Robyn, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. *LuminosoInsight/wordfreq: v2.2.* doi:10.5281/zenodo.1443582.

Srinivasan, Mahesh, and Hugh Rabagliati. 2015. ”How concepts and conventions structure the lexicon: Cross-linguistic evidence from polysemy”. *Lingua* 157:124–152. ISSN: 00243841. doi:10.1016/j.lingua.2014.12.004.

Sweetser, Eve. 1990. *From Etymology to Pragmatics: Metaphorical and Cultural Aspects of Semantic Structure.* Cambridge Studies in Linguistics v. 54. Cambridge University Press. ISBN: 9780521324069. doi:10.1017/CBO9780511620904.

Taghipour, Kaveh, and Hwee Tou Ng. 2015. ”One Million Sense-Tagged Instances for Word Sense Disambiguation and Induction”. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning,* 338–344. Beijing, China: Association for Computational Linguistics. doi:10.18653/v1/K15-1037.

Tarhio, Jorma, Jan Holub, and Emanuele Giaquinta. 2017. ”Technology beats algorithms (in exact string matching)”. *Software: Practice and Experience* 47 (12): 1877–1885. doi:10.1002/spe.2511.

Thomason, Sarah G. 2001. ”Contact-Induced Language Change: Results”. In *Language Contact: An Introduction.* Edinburgh University Press. ISBN: 0748607196.

Tiedemann, Jörg. 2012. ”Parallel Data, Tools and Interfaces in OPUS.” In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12),* 2012:2214–2218. Istanbul, Turkey: European Language Resources Association (ELRA). https://www.aclweb.org/anthology/L12-1246.

Truscott, John. 1998. ”Noticing in second language acquisition: a critical review”. *Second Language Research* 14 (2): 103–135. doi:10.1191/026765898674803209. eprint: http://sdkrashen.com/content/articles/noticing{_}1998.pdf.

Turing, Alan M. 1950. ”Computing machinery and intelligence”. *Mind* 59 (236): 433–460. doi:10.1093/mind/LIX.236.433.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. ”Attention is all you need”. In *Advances in Neural Information Processing Systems,* 5999–6009. Curran Associates, Inc. arXiv: 1706.03762. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Véronis, Jean. 2004. ”HyperLex: lexical cartography for information retrieval”. *Computer Speech & Language* 18 (3): 223–252. ISSN: 0885-2308. doi:10.1016/j.csl.2004.05.002.

Vervaet, Ruth. 2017. ”English loanwords in the Chinese lexicon”. Master's thesis, Ghent University. https://lib.ugent.be/catalog/rug01:002349097.

Vial, Loïc, Benjamin Lecouteux, and Didier Schwab. 2019. ”Sense Vocabulary Compression through the Semantic Knowledge of WordNet for Neural Word Sense Disambiguation”. In *Proceedings of the 10th Global Wordnet Conference.* Wroclaw, Poland. arXiv: 1905.05677.

Vinh, Nguyen Xuan, Julien Epps, and James Bailey. 2009. ”Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?” In *Proceedings of the 26th Annual International Conference on Machine Learning,* 1073–1080. ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery. ISBN: 9781605585161. doi:10.1145/1553374.1553511.

Virpioja, Sami, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. *Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline.* Report, Aalto University publication series SCIENCE + TECHNOLOGY 25. Helsinki, Finland: Department of Signal Processing and Acoustics, Aalto University. http://urn.fi/URN:ISBN:978-952-60-5501-5.

Wang, Shan, and Francis Bond. 2013. ”Building the Chinese Open Wordnet (COW): Starting from Core Synsets”. In *Proceedings of the 11th Workshop on Asian Language Resources,* 10–18. Nagoya, Japan: Asian Federation of Natural Language Processing. https://www.aclweb.org/anthology/W13-4302.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, et al. 2016. ”Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. *CoRR.* arXiv: 1609.08144.

Yimam, Seid Muhie, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. ”WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations”. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations,* 1–6. Sofia, Bulgaria: Association for Computational Linguistics. `https://www.aclweb.org/anthology/P13-4001`.

Zhong, Zhi, and Hwee Tou Ng. 2010. ”It makes sense: A wide-coverage word sense disambiguation system for free text”. In *Proceedings of the ACL 2010 system demonstrations,* 78–83. Association for Computational Linguistics.

Zilio, Leonardo, Rodrigo Wilkens, and Cédrick Fairon. 2017. ”Using NLP for Enhancing Second Language Acquisition”: 839–846. ISSN: 13138502. doi:`10.26615/978-954-452-049-6_107`.